RED4SEC

Non-Plagiarism Audit Report

Swingby Protocol

07-02-2020

Contents

1. Executive Summary 3
  1.1 Motivations 3
  1.2 Coverage and Purpose 4
2. Projects Overview 5
  2.1 Swingby 5
  2.2 Thorchain 5
  2.3 Conclusions 5
3. Non-Plagiarism Statement 7
  3.1 Licenses 7
  3.2 Whitepaper and Documentation Analysis 9
  3.3 Source Code Implementation Review 10
4. Conclusions 19

1. Executive Summary

In recent years, the emergence of thousands of cryptocurrency startups and blockchain-based projects has resulted in an increase of scandals and accusations concerning plagiarism.

Plagiarism is one of the main growing global problems experienced by developers and researchers, especially in spheres where innovation is essential where even successful and legitimate projects are exposed to accusations of plagiarism.

In a world that is committed to the innovation of these disruptive blockchain-based technologies, it is imperative to be clear and concise with investors and customers before they decide to adopt a product that could result in being be misleading or fraudulent.

On January 15, 2020, *Malcom Lerider*, Research and Development Advisor and *Yusaku Senga*, CEO and Technical Architect of Swingby protocol, contacted *Red4Sec Cybersecurity* to perform a non-plagiarism audit given the recent accusations received by members of other crypto communities in social networks.

1.1 Motivations

One of the main reasons and motivations for this non-plagiarism audit is the recent accusations of plagiarism on social networks by few people in the blockchain community.

In order to analyze if there could indeed be any sign of plagiarism, we decided to compare the Swingby code with one of the projects currently online, which is more similar in terms of concept to Swingby, this chosen project is Thorchain[1].

Red4Sec audit team will analyze both project's source code responsibly and impartially in order to determine if there is evidence of plagiarism in the proposed solutions, ideas, whitepaper or source code implementation.

[1] https://docs.thorchain.org

1.2 Coverage and Purpose

As requested by Swingby, Red4Sec Cybersecurity has been asked to perform an independent non-plagiarism audit over its blockchain-based project, evaluating all the project documentation, specifications and source code implementations.

This performed audit procedure is mainly designed to evaluate the legitimacy, authenticity and originality of Swingby protocol project.

After the rigorous and impartial technical/theoretical evaluation of all the projects involved in the accusations, Red4Sec will present their final conclusions in order to provide investors and new users of the project with our opinion on this with the greatest possible transparency.

The non-plagiarism audit was conducted from 3rd February 2020 to 7th February 2020 by Red4Sec's blockchain specialists and technical auditors.

2. Projects Overview

Before starting the analysis, we will make an overview of the Swingby and Thorchain projects in order to understand the purpose of both projects and the solutions they provide to the crypto ecosystem.

This section will provide a general overview of both projects which will facilitate subsequent non-plagiarism evaluation and the audit.

2.1 Swingby

The Swingby Network is a permissionless, peer-to-peer network of nodes who run the Swingby node software to communicate with one another.

Swingby Skybridge brings fast and non-custodial token swaps to all ECDSA-based blockchains using modern threshold signature cryptography and an open source layer 2 peer-to-peer staking node implementation. Bitcoin, Ethereum, Binance Chain, EOS, Tron, Ripple, Dash, and many more blockchains are 100% compatible.

2.2 Thorchain

Thorchain is a lightning fast decentralized liquidity network which facilitates continuous cross-chain liquidity pools with no pegged tokens. It has a fee incentive system that has been specifically designed for its increasingly attractive on-chain liquidity.

Thorchain uses pBFT consensus to achieve sub-second block finality and tokens are traded on single chains. Its native protocol facilitates on-chain atomic asset swaps at the protocol level.

Thorchain solves the fundamental problems of asset swaps with on-chain continuous liquidity and correct incentivization economics for security.

2.3 Conclusions

Fundamentally, Swingby Skybridge aims to address the need for a simple token bridge to/from Binance Chain, Bitcoin and other blockchains to increase trading volume and allow users to take advantage of the features of other platforms in the crypto ecosystem.

Skybridge is a "layer 2" protocol that is built on top of existing base blockchain systems (layer 1) while Thorchain, a popular project that aims to facilitate inter-blockchain token swaps using liquidity pools similar to Uniswap[2], relies on a Cosmos-based "state chain" that functions more like a layer 1 sidechain than a layer 2 solution.

Both projects share certain similarities, with both of them providing solutions to the liquidity cross-chain challenge for swap between assets across chains. Also, both projects use the Threshold Signature Scheme (TSS), a protocol where private keys, and therefore cryptocurrency addresses, can be created by multiple parties (described by Rosario Gennaro and Steven Goldfeder[3]). However, its technical implementation and code design are completely different.

The Swingby network implements two important processes using TSS Protocol. First, the keygen process, which is the collaborative creation of a public key, from which custodial cryptocurrency addresses on both blockchains are derived. And second, the transaction signing process which is the collaborative signing of cryptocurrency transactions for making payments from those custodial addresses.

Nevertheless, the multi-sig (TSS) elements of Thorchain are implemented as helper scripts that form a P2P network outside the state chain itself, meaning that the P2P network for TSS and the Tendermint[4] P2P network that connects the blockchain nodes together are different pieces of software.

[2] https://uniswap.io
[3] https://eprint.iacr.org/2019/114.pdf
[4] https://tendermint.com

3. Non-Plagiarism Statement

Plagiarism can be considered as one of the major types of scientific misconduct and is defined as;

> *"Taking over the ideas, methods, implementations or techniques of another, without acknowledgment and with the intention that they be taken as the work of the deceiver".*[5]

In this report, we will analyze all possible ways in which plagiarism can manifest itself. It should be noted that source code plagiarism could vary from copy-pasting small source code amounts to plagiarizing large chunks of source code and masking everything with some deception techniques to disguise copied codes.

The level of sophistication of these plagiarism techniques will be analyzed based on papers presented by *J.A.W. Faidhi and S.K. Robinson*[6]. These investigations analyze different aspects within the plagiarism of code such as,

1. Comments and indentation.
2. Identifiers and declarations.
3. Program modules and code implementations.
4. Program statements and decision logic.

3.1 Licenses

Before evaluating the similarities in the documentation, in the whitepaper, and in the source code implementation, it is essential to know the licenses assigned to each of the projects involved in the accusations and controversies; more specifically, ***Thorchain*** and ***Swingby****.*

Does the distribution contain a copy of the license? Is it clearly stated what software is covered by the license? Does the license include the distribution and use by third parties of the source code? - These are some of the questions that should be answered when analyzing the licenses of both projects.

[5] https://www.lynn.edu/university-policies/vol-iii-academic-policies/academic-dishonesty-and-plagiarism-by-faculty-policy
[6] https://www.sciencedirect.com/science/article/abs/pii/036013158790042X

In this section, Red4Sec will carry out a small study of the licenses of each project and its components, to later know if any of them are violated and what their implications are.
In any case, this is a superficial analysis of the licenses and, if necessary, a detailed study should be carried out by experts in legality, patents and intellectual property.

Thorchain

After a broad search on Thorchain project code licenses, it has been verified that both their Github and Gitlab repositories[7] refers to MIT[8] and GPL v3.0 licenses.

- https://gitlab.com/thorchain/thornode/-/blob/master/LICENSE
- https://gitlab.com/thorchain/thorchain-sites/thorchain-org/-/blob/master/LICENSE
- https://github.com/thorchain/THORChain/blob/master/LICENSE.md
- https://github.com/thorchain/THORChain.info/blob/master/LICENSE
- https://github.com/thorchain/Resources/blob/master/LICENSE.md

Additionally, the project owners themselves have publicly stated[9] that the project is a copy-free opensource project.

> *"THORChain is an open-source and public project. Anyone is free to copy it at will. The project with the greatest purchasing power will consume the others."*

This statement does not evidence or ensure that the Swingby protocol has used or partially plagiarized parts of the Thorchain code, it simply exempts Swingby or any other project from any responsibility or accusation of plagiarism.

Swingby

At the time of this audit, Swingby source code is not publicly accessible, so no license has yet been published. However, it will be licensed by AGPL 3.0[10].

[7] https://gitlab.com/thorchain/
[8] https://opensource.org/licenses/MIT
[9] https://twitter.com/thorchain_org/status/1212355574144192512
[10] https://www.gnu.org/licenses/agpl-3.0.en.html

Since the source code is not public, Red4Sec has signed a *Non-Disclosure Agreement* to be able to access the source code[11] and proceed with its review.

As a conclusion at this point, neither of the licenses of both projects prevent or protect, but rather promote the use of personalization or work derived from both projects, as they are mainly based on the MIT and open source license.

3.2 Whitepaper and Documentation Analysis

In this section, Red4Sec has analyzed each project and its documentation in order to describe the ideas, objectives and solutions implemented by each one.

Swingby[12] and **Thorchain**[13] whitepapers have been analyzed and compared to look for possible similarities or any evidence of plagiarism.

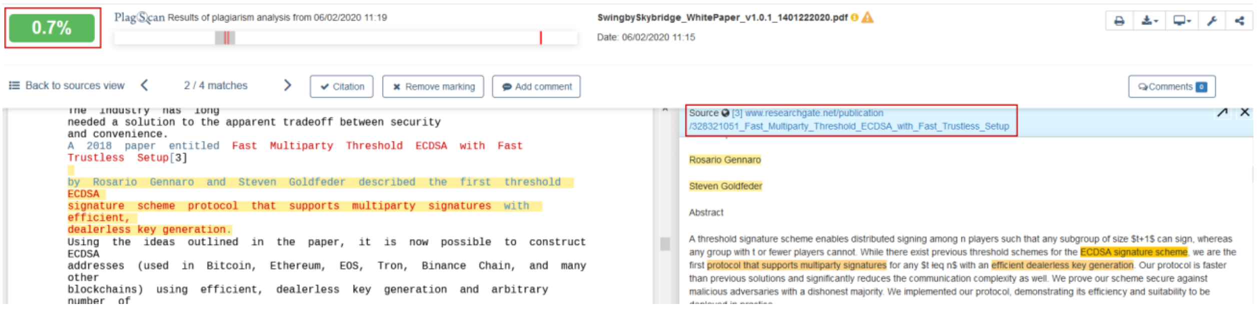
In addition to the manual analysis and specifications review of the whitepapers, we have decided to include the results of one of the most famous plagiarism detection tools, PlagScan[14].

PlagScan establishes the plagiarism percentage of documents based on the number of references between them and against public sources on the Internet.

A reference whitepaper has been chosen to assess whether the number of matches is significant compared to any other whitepaper. The project chosen for this purpose is "Bitcoin: A Peer-to-Peer Electronic Cash System"[15], the original Bitcoin's whitepaper.

Once the Swingby whitepaper has been compared against Thorchain documentation and the different results[16] analyzed, the percentage of plagiarism was **0.7%**.

[11] SHA256 HASH *1AD1A968B6C4C17156B61B414EE7CDBD57EE84C07AF250F6388319F101E86299*
[12] https://docs.swingby.network/SwingbySkybridge_WhitePaper_v1.0.1_1401222020.pdf
[13] https://github.com/thorchain/Resources/tree/master/Whitepapers/THORChain
[14] https://www.plagscan.com
[15] https://bitcoin.org/bitcoin.pdf
[16] https://www.plagscan.com/doc?129304725&sharekey=l60aOGnlKjkVwKo6aP0u



PlagScan Results of plagiarism analysis from 06/02/2020 11:19

SwingbySkybridge_WhitePaper_v1.0.1_1401222020.pdf

Date: 06/02/2020 11:15

```
The  industry  has  long
needed a solution to the apparent tradeoff between security
and convenience.
A  2018  paper  entitled  Fast  Multiparty  Threshold  ECDSA  with  Fast
Trustless  Setup[3]

by  Rosario  Gennaro  and  Steven  Goldfeder  described  the  first  threshold
ECDSA
signature  scheme  protocol  that  supports  multiparty  signatures  with
efficient,
dealerless key generation.
Using  the  ideas  outlined  in  the  paper,  it  is  now  possible  to  construct
ECDSA
addresses  (used  in  Bitcoin,  Ethereum,  EOS,  Tron,  Binance  Chain,  and  many
other
blockchains)  using  efficient,  dealerless  key  generation  and  arbitrary
number  of
```

Source [3] www.researchgate.net/publication/328321051_Fast_Multiparty_Threshold_ECDSA_with_Fast_Trustless_Setup

Rosario Gennaro

Steven Goldfeder

Abstract

A threshold signature scheme enables distributed signing among n players such that any subgroup of size $t+1$ can sign, whereas any group with t or fewer players cannot. While there exist previous threshold schemes for the ECDSA signature scheme, we are the first protocol that supports multiparty signatures for any $t \leq n$ with an efficient dealerless key generation. Our protocol is faster than previous solutions and significantly reduces the communication complexity as well. We prove our scheme secure against malicious adversaries with a dishonest majority. We implemented our protocol, demonstrating its efficiency and suitability to be

The only coincidence is the reference to "Fast Multiparty Threshold ECDSA with Fast by Rosario Gennaro and Steven Goldfeder"[17], which can clearly be ruled out as a reference to the technical paper of the technology used, which is correctly referenced in the Swingby whitepaper.

After evaluating the exposed evidences, it can be concluded that there are no references between both whitepapers, nor with any other projects related to the blockchain world; all the references found are totally legitimate and justified.

3.3 Source Code Implementation Review

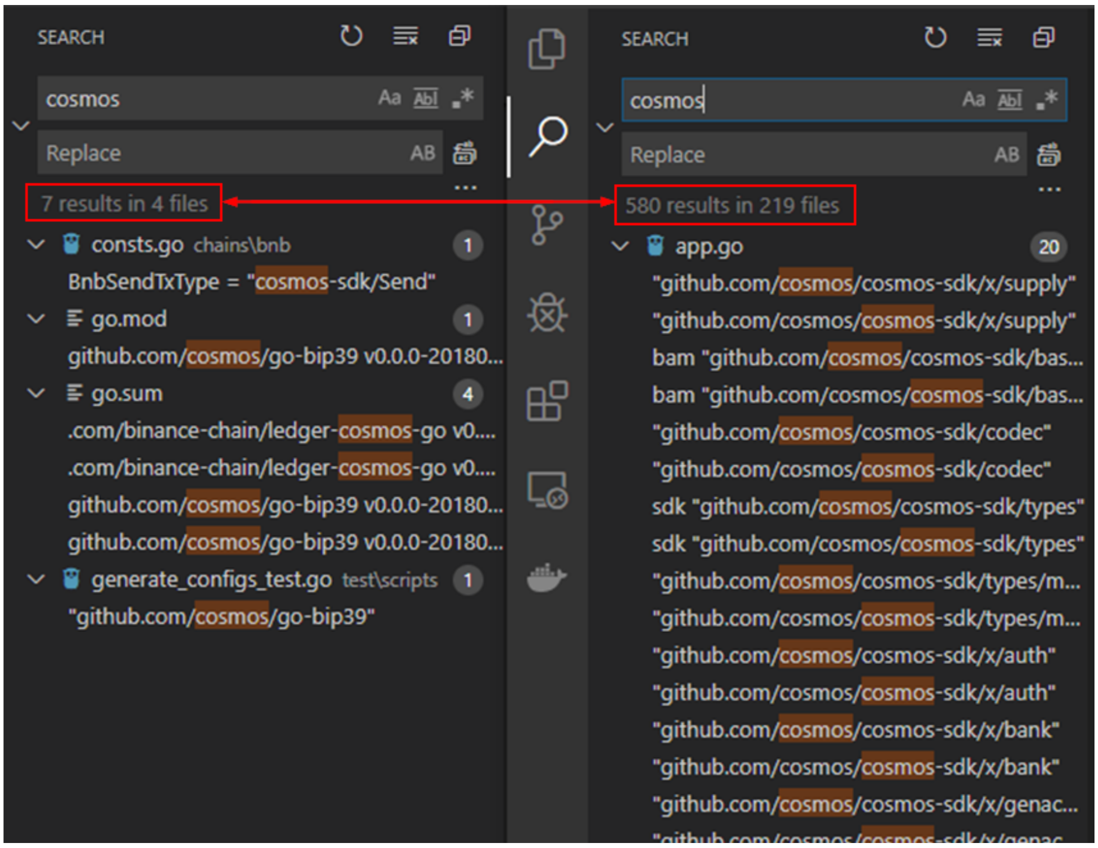
In this section, Red4Sec will make a comparison between both projects source code in order to determine the similarity and the importance of similarities. This is a point to consider since, for example, similarities in the connection to a server are not as equally important as the implementation of a compression system or distribution mechanism.

[17] https://eprint.iacr.org/2019/114.pdf

The technical team has performed a comprehensive review of the code, and has detected some technical and implementation-level points that demonstrate that the code differs from the Thorchain code.

- Swingby uses Protobuffer for network communications.
- Totally different files documentation and comments along the code.
- Main functions source code is totally different.

Before an in-depth analysis, as **Thorchain** is based on Cosmos[18], a quick search for 'cosmos' on both projects should denote similarities in the case of plagiarism, however the following evidence indicates the opposite.



```
SEARCH
cosmos
Replace
7 results in 4 files
consts.go chains\bnb                                  1
  BnbSendTxType = "cosmos-sdk/Send"
go.mod                                                1
  github.com/cosmos/go-bip39 v0.0.0-20180...
go.sum                                                4
  .com/binance-chain/ledger-cosmos-go v0....
  .com/binance-chain/ledger-cosmos-go v0....
  github.com/cosmos/go-bip39 v0.0.0-20180...
  github.com/cosmos/go-bip39 v0.0.0-20180...
generate_configs_test.go test\scripts                 1
  "github.com/cosmos/go-bip39"
```

```
SEARCH
cosmos
Replace
580 results in 219 files
app.go                                               20
  "github.com/cosmos/cosmos-sdk/x/supply"
  "github.com/cosmos/cosmos-sdk/x/supply"
  bam "github.com/cosmos/cosmos-sdk/bas...
  bam "github.com/cosmos/cosmos-sdk/bas...
  "github.com/cosmos/cosmos-sdk/codec"
  "github.com/cosmos/cosmos-sdk/codec"
  sdk "github.com/cosmos/cosmos-sdk/types"
  sdk "github.com/cosmos/cosmos-sdk/types"
  "github.com/cosmos/cosmos-sdk/types/m...
  "github.com/cosmos/cosmos-sdk/types/m...
  "github.com/cosmos/cosmos-sdk/x/auth"
  "github.com/cosmos/cosmos-sdk/x/auth"
  "github.com/cosmos/cosmos-sdk/x/bank"
  "github.com/cosmos/cosmos-sdk/x/bank"
  "github.com/cosmos/cosmos-sdk/x/genac...
  "github.com/cosmos/cosmos-sdk/x/genac
```

Swingby / Thorchain

Subsequently, a deeper analysis was carried out based on certain criteria that, in the case that it was positive, it would have indicated that it is indeed a plagiarism.

[18] https://cosmos.network

The name of the files, comments in the code, implementation of main functions, string values and numerical values are some of the elements reviewed in the code. In order to speed up the analysis of strings and integers, Red4Sec has used a specific software that analyzes the syntax of Golang files using *ANTLR4*[19] . This has been done to create the Syntax Tree[20] of the projects and give us the ability to compare projects ignoring the different formats, names of variables and classes, plus any other object that does not affect the logic of the code.

A reference project has been chosen to assess whether the number of matches is significant compared to any other software. The project chosen for this purpose is **btcd**[21], a full node bitcoin implementation written in Go.

Analysis Results

Swingby	**Thorchain**	**Btcd**
https://github.com/SwingbyProtocol/swapd-go/tree/master	*https://gitlab.com/thorchain/thornode/-/tree/master*	*https://github.com/btcsuite/btcd/tree/master*
Against **Thorchain**	Against **Btcd**	Against **Swingby**
File names: File names are the same in clearly generalized cases such as *types.go* or project files. There is no noticeable difference with respect to a random bitcoin node in Go.		
Files: 149 Similarities: 15 Coincidence: 10.07%	Files: 395 Similarities: 17 Coincidence: 4.3%	Files: 324 Similarities: 16 Coincidence: 4.94%
.gitignore broadcast.go config.go dockerfile go.mod go.sum keeper.go main.go makefile readme.md stake.go	.gitignore chain.go chain_test.go config.go generate.go genesis.go go.mod go.sum helpers.go helpers_test.go license	.gitignore compress.go config.go config_test.go doc.go error.go go.mod go.sum main.go params.go peer.go

[19] https://github.com/antlr/grammars-v4/tree/master/golang
[20] https://en.wikipedia.org/wiki/Abstract_syntax_tree
[21] https://github.com/btcsuite/btcd

stake_test.go swap.go symbol.go types.go	main.go pubkey.go pubkey_test.go readme.md sign.go sign_test.go	protocol.go readme.md server.go utils.go version.go
Comments: As we can see, there are more similarities when compared with a bitcoin node than with the Thorchain itself.		
Comments: 744 Similarities: 2 Coincidence: 0.27%	Comments: 1113 Similarities: 6 Coincidence: 0.54%	Comments: 16630 Similarities: 6 Coincidence: 0.04%

//
// indirect
	//
	// 1
	// Bitcoin core only supports HTTP POST mode
	// indirect
	// it.
	// or not.
		//
		// and we mirror that too.
		// indirect
		// Nonce
		// OpenSSL right shifts excess bits from the number if the hash is too large
		// Version

Imports: There is no import that indicates any plagiarism, since all are necessary for the purpose of the project, and any similarity is not far from other projects, such as btcd.		
Imports: 126 Similarities: 43 Coincidence: 34.13%	Imports: 122 Similarities: 30 Coincidence: 24.59%	Imports: 101 Similarities: 42 Coincidence: 41.58%
"bytes" "context" "crypto/sha256" "crypto/tls" "encoding/base64" "encoding/binary" "encoding/hex" "encoding/json" "errors" "flag" "fmt" "github.com/binance-chain/go-sdk/client" "github.com/binance-chain/go-sdk/common/types" "github.com/binance-chain/go-sdk/keys" "github.com/binance-chain/go-sdk/types" "github.com/binance-chain/go-	"bufio" "bytes" "crypto/sha256" "crypto/tls" "encoding/base64" "encoding/binary" "encoding/hex" "encoding/json" "errors" "fmt" "io" "io/ioutil" "log" "math/big" "math/rand" "net/http" "net/url" "os" "os/signal" "path/filepath"	"bytes" "container/list" "crypto/cipher" "crypto/elliptic" "crypto/sha256" "crypto/sha512" "crypto/tls" "crypto/x509" "encoding/base64" "encoding/binary" "encoding/hex" "encoding/json" "errors" "fmt" "github.com/btcsuite/btcd/btcec" "github.com/btcsuite/btcd/chaincfg" "github.com/btcsuite/btcd/chaincfg/chainhash" "github.com/btcsuite/btcd/txscript" "github.com/btcsuite/btcd/wire"

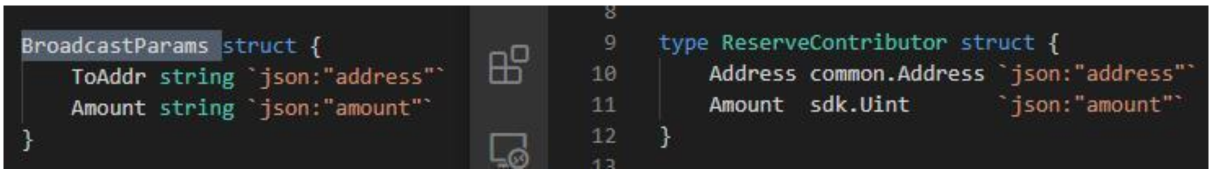
sdk/types/msg" "github.com/binance-chain/go-sdk/types/tx" "github.com/btcsuite/btcutil/bech32" "github.com/pkg/errors" "github.com/spf13/pflag" "github.com/spf13/viper" "github.com/tendermint/tendermint/crypto" "github.com/tendermint/tendermint/crypto/secp256k1" "io" "io/ioutil" "math/big" "math/rand" "net/http" "net/url" "os" "os/signal" "path" "path/filepath" "reflect" "regexp" "sort" "strconv" "strings" "sync" "sync/atomic" "syscall" "testing" "time"	"reflect" "regexp" "sort" "strconv" "strings" "sync" "sync/atomic" "syscall" "testing" "time"	"github.com/btcsuite/btcutil" "io" "io/ioutil" "math" "math/big" "math/rand" "net" "net/http" "net/url" "os" "os/signal" "path/filepath" "reflect" "regexp" "runtime" "sort" "strconv" "strings" "sync" "sync/atomic" "syscall" "testing" "time"
Numerics: When comparing the syntax tree in search of numerical values, you can see that there are more similarities with a random bitcoin node than against Thorchain.		
Numerics: 314 Similarities: 34 Coincidence: 10.83%	Numerics: 183 Similarities: 85 Coincidence: 46.45%	Numerics: 801 Similarities: 297 Coincidence: 37.08%
0,0644,1,10,100,1000,100000,1000000,1024,11,12,1440,15,150,2,20,200,25,255,256,3,30,32,4,400,5,50,500,6,60,64,7,720,8	0,0644,1,10,100,1000,10000,100000,1000000,10000000,100000000,101,1014,1023,1024,103,105,107,108,11,110,111,115,12,120,123,123456789,125,13,133,134,14,15,150,167,17,18,19,2,20,200,2000,20000000,201,23,25,255,256,27,3,30,300,3000,3000000,32,33,34,35,37,4,40,400,4000,400000,45,5,50,500,5000,50000,55,6,60,64,66,67,7,72,75,79,8,9,90,9223372036854775807,99	0,0644,0755,0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d,0x1e,0x1f,0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3a,0x3b,0x3c,0x3d,0x3e,0x3f,0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4a,0x4b,0x4c,0x4d,0x4e,0x4f,0x50,0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,0x5a,0x5b,0x5c,0x5d,0x5e,0x5f,0x60,0x61,0x62,0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x6a,0x6b,0x6c,0x6d,0x6e,0x6f,0x70,0x71,0x72,

		0x73,0x74,0x75,0x76,0x77,0x78,0x7a,0x7b,0x7c,0x7d,0x7e,0x7f,0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8a,0x8b,0x8c,0x8d,0x8e,0x8f,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9a,0x9b,0x9c,0x9d,0x9e,0x9f,0xa0,0xa1,0xa2,0xa3,0xa4,0xa5,0xa6,0xa7,0xa8,0xa9,0xaa,0xab,0xac,0xad,0xae,0xaf,0xb1,0xb2,0xb4,0xb5,0xb6,0xb7,0xb8,0xb9,0xba,0xbb,0xbc,0xbd,0xbe,0xbf,0xc0,0xc1,0xc2,0xc3,0xc4,0xc5,0xc6,0xc7,0xc8,0xc9,0xca,0xcb,0xcc,0xcd,0xce,0xcf,0xd0,0xd1,0xd2,0xd3,0xd4,0xd5,0xd6,0xd7,0xd8,0xd9,0xda,0xdb,0xdc,0xdd,0xde,0xdf,0xe0,0xe1,0xe2,0xe3,0xe4,0xe5,0xe6,0xe7,0xe8,0xe9,0xea,0xec,0xed,0xee,0xef,0xf0,0xf1,0xf2,0xf3,0xf4,0xf5,0xf6,0xf7,0xf8,0xf9,0xfa,0xfb,0xfc,0xfd,0xfe,0xff,1,10,100,1000,100000,1000000,1024,11,12,1234,127,128,1337,15,150,16,2,20,200,25,250,255,256,28,299,3,30,31,32,4,400,5,50,500,512,58,6,60,61,64,7,8
Strings: When comparing the syntax tree in search of string values, there are only similarities clearly related to blockchain technologies, no clear allusion to possible plagiarism.		
Strings: 1257 Similarities: 108 Coincidence: 8.59%	Strings: 2398 Similarities: 114 Coincidence: 4.75%	Strings: 8646 Similarities: 103 Coincidence: 1.19%
" " "" "-" "%d" "%s %s" "%s.%s" "%s/%s" "%v" ", " "," "." "/" ":%d" "\n" "0" "0.1.0" "127.0.0.1" "address" "application/json" "bnb" "BNB" "BTC" "bytes" "c"	" " " " "" "-" "%" "%d" "%s" "%s%s" "%s:%d" "%s\n" "%v" "*" ", " "," "." "/" ":" "\n" "_" "=" "0" "0000000000000000000000000000000000000000000000000000000000000000"	" " "" "-" "%d" "%s:%s" "%v" ", " "," "." "/" "\n" "|" "0" "1" "1.0" "100" "127.0.0.1" "address" "application/json" "bytes" "c" "container/list" "crypto/cipher" "crypto/elliptic"

"config"	"0x"	"crypto/sha256"
"context"	"1.3.0"	"crypto/sha512"
"crypto/sha256"	"10"	"crypto/tls"
"crypto/tls"	"127.0.0.1"	"crypto/x509"
"encoding/base64"	"2"	"data"
"encoding/binary"	"5"	"encoding/base64"
"encoding/hex"	"a"	"encoding/binary"
"encoding/json"	"abcd"	"encoding/hex"
"errors"	"active"	"encoding/json"
"f"	"add"	"ERROR"
"flag"	"address"	"errors"
"fmt"	"application/json"	"f"
"github.com/binance-chain/go-sdk/client"	"b"	"fmt"
"github.com/binance-chain/go-sdk/common/types"	"block"	"github.com/btcsuite/btcd/btcec"
"github.com/binance-chain/go-sdk/keys"	"bogus"	"github.com/btcsuite/btcd/chaincfg"
"github.com/binance-chain/go-sdk/types"	"bufio"	"github.com/btcsuite/btcd/chaincfg/chainhash"
"github.com/binance-chain/go-sdk/types/msg"	"bytes"	"github.com/btcsuite/btcd/txscript"
"github.com/binance-chain/go-sdk/types/tx"	"c"	"github.com/btcsuite/btcd/wire"
"github.com/btcsuite/btcutil/bech32"	"Content-Type"	"github.com/btcsuite/btcutil"
"github.com/pkg/errors"	"crypto/sha256"	"http"
"github.com/spf13/pflag"	"crypto/tls"	"https"
"github.com/spf13/viper"	"d"	"int64"
"github.com/tendermint/tendermint/crypto"	"e"	"io"
"github.com/tendermint/tendermint/crypto/secp256k1"	"encoding/base64"	"io/ioutil"
"http"	"encoding/binary"	"localhost"
"https"	"encoding/hex"	"math"
"io"	"encoding/json"	"math/big"
"io/ioutil"	"error"	"math/rand"
"k"	"errors"	"message"
"keygen"	"f"	"n"
"l"	"failed"	"net"
"localhost"	"fmt"	"net/http"
"m"	"foo"	"net/url"
"math/big"	"from"	"NONE"
"math/rand"	"g"	"os"
"n"	"hash"	"os/signal"
"net/http"	"http"	"path/filepath"
"net/url"	"https"	"POST"
"not exist"	"info"	"received"
"os"	"io"	"reflect"
"os/signal"	"io/ioutil"	"regexp"
"p"	"local"	"runtime"
"page"	"localhost"	"send"
"path"	"log"	"sort"
"path/filepath"	"math/big"	"strconv"
"query"	"math/rand"	"strings"
"reflect"	"mocknet"	"sync"
"regexp"	"n"	"sync/atomic"
"rewards"	"net/http"	"syscall"
"sort"	"net/url"	"t"
"stake"	"not implemented"	"tcp"
	"os"	"test"
	"os/signal"	"testing"
	"outbound"	"time"
	"path/filepath"	"type"
	"payload"	"value"
	"pubkey"	`json:"address"`
	"reflect"	`json:"address,omitempty"`
	"regexp"	`json:"amount"`
	"s"	

"status"	"sort"	`json:"asm"`
"strconv"	"strconv"	`json:"code,omitempty"`
"strings"	"strings"	`json:"fee,omitempty"`
"sync"	"sync"	`json:"hash"`
"sync/atomic"	"sync/atomic"	`json:"hash,omitempty"`
"syscall"	"syscall"	`json:"height"`
"t"	"t"	`json:"hex"`
"tbnb"	"test"	`json:"locktime"`
"test"	"testing"	`json:"mediantime"`
"testing"	"testnet"	`json:"n"`
"testnet-dex.binance.org"	"time"	`json:"params"`
"time"	"true"	`json:"result"`
`json:"-"`	"tx"	`json:"sequence"`
`json:"address"`	"unknown"	`json:"txid"`
`json:"amount"`	"version"	`json:"type"`
`json:"blockHeight"`	`"%s"`	`json:"value"`
`json:"code"`	`json:"address"`	`json:"version"`
`json:"hash"`	`json:"amount"`	`json:"vin"`
`json:"height"`	`json:"chain"`	`json:"vout"`
`json:"log"`	`json:"data"`	`json:"weight"`
`json:"memo"`	`json:"f"`	
`json:"message"`	`json:"fee"`	
`json:"msg"`	`json:"hash"`	
`json:"result"`	`json:"height"`	
`json:"s"`	`json:"id"`	
`json:"sequence"`	`json:"jsonrpc"`	
`json:"t"`	`json:"result"`	
`json:"to"`	`json:"sequence"`	
`json:"txArray"`	`json:"status"`	
`json:"txs"`	`json:"type"`	
`json:"type"`	`json:"value"`	
`json:"value"`	`json:"version"`	
`json:"version"`		

After analyzing the results, even both projects are developed in *Go*[22], the only notable coincidence is the following structure, which can clearly be ruled out as it is not an important part of the project's logic or code implementation.



```
BroadcastParams struct {
    ToAddr string `json:"address"`
    Amount string `json:"amount"`
}
```

```
type ReserveContributor struct {
    Address common.Address `json:"address"`
    Amount  sdk.Uint       `json:"amount"`
}
```

- Swingby: https://github.com/SwingbyProtocol/swapd-go/blob/a12fe94f7ec8710e93d240811de42968dee0636f/common/types.go#L71-L74

[22] https://golang.org

- Thorchain: https://gitlab.com/thorchain/thornode/-/blob/master/x/thorchain/types/type_reserve_contributor.go#L9-12

4. Conclusions

After analyzing the Swingby's project implementation, protocol and design in depth, and after comparing it with other projects with similar characteristics, as of today, 7th February 2020, Red4Sec is firmly convinced that this project is an innovative, self-made and plagiarism-free project.

As we have seen in the previous sections, both projects, despite having similar solutions, differ completely in their implementation and are more like a third project, *btcd*, than each other.

Blockchain technologies that have awakened in recent years are mostly open source projects that feed each other to build better technologies and improve current implementations.

For this reason, we cannot speak of plagiarism when the philosophy that accompanies the development of these blockchain-based projects is that of sharing and exchanging ideas and solutions, as well as creating transparent and reliable technologies.

Similarities between projects allows for the creation of synergies to obtain advantages and provide added value to the blockchain community.

RED4SEC

Invest in Security, invest in your future