Audit Report

Sifchain

Cryptographic Analysis

Jul 30, 2021

Table of Contents

License

Disclaimer

This audit has been performed by

Iraklis Leontiadis

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of this Report

Oak Security has been engaged by Black Hole Industries to perform a continuous security audit of the Sifchain blockchain implementation. This current audit report covers a review of the use of cryptographic libraries and primitives in the Sifchain node implementation.

The objectives of the audit are as follows:

1. Determine the correct use of cryptography.
2. Determine that cryptographic libraries are used correctly.
3. Determine that any cryptographic systems are initialized with the correct parameters.
4. Analyze dependencies of cryptographic protocol implementations.
5. Make recommendations to improve cryptographic security.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following GitHub repository:

https://github.com/Sifchain/sifnode/tree/feature/cosmos-0.42

Commit hash: `592388316f90b18492871664adf36b0df39b1d53`

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Manual review of cryptographic code
3. Report preparation

Summary Overview

The submitted codebase does not implement new cryptographic algorithms and uses only external libraries. We found some lack of API documentation for the underlying cosmos SDK which may malfunction the entire protocol due to lack of clarity. The questions we tried to address while reviewing the codebase are:

1. Existence of external cryptographic libraries
2. New cryptographic code implementation
3. Sound tests
4. Crypto flaws in the code

The code uses the following external libraries:

1. go libs: `crypto,crypto/ecdsa`
2. github modules:
    a. `github.com/ethereum/go-ethereum/crypto`
    b. `github.com/miguelmota/go-solidity-sha3`
    c. `github.com/cosmos/cosmos-sdk/crypto/keys`
    d. `github.com/cosmos/cosmos-sdk/crypto/keys/hd`
    e. `github.com/tyler-smith/go-bip39`

and lies on the secure implementation thereof. We did not perform any audit on them. The external libraries are used for key derivation and signing of byte streams. It is assumed secure entropy extraction and secure key storage. The external libraries implement:

1. SHA256, SHA3, PBKDF2, ECDSA with sec256k1
2. the bip39 proposal for mnemonic phrase derivation

No serious flaws have been found. SHA256 can be replaced with SHA3 for better state of the art usage. ECDSA has no security flaws but the alternative of Ed25519 provides better resistance to side channel attacks and security in case of a broken RNG.

How to read this Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending, Acknowledged** or **Resolved**. Informational notes do not have a status, since we consider them optional recommendations.

Note, that audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note, that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Summary of Findings

No	Description	Severity	Status
1	empty password input	Minor	Pending
2	weak access control	Minor	Pending
3	adopt state of the art	Informational	

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	-
Level of Documentation	Medium	- High level doc is great: https://docs.sifchain.finance/ but code needs some more updated documentation for better going through code
Test Coverage	Medium	- Test coverage can be expanded with fuzzing tools as https://github.com/google/gofuzz

Detailed Findings

1. A minor issue

Severity: Minor

https://github.com/Sifchain/sifnode/blob/bebbe9883560bbde4f452f81a2d85bdbc243636a/cmd/ebrelayer/relayer/ethereum.go#L68

https://github.com/Sifchain/sifnode/blob/bebbe9883560bbde4f452f81a2d85bdbc243636a/tools/sifgen/key/key.go#L44

The third parameter `bip39Passwd` is empty and it is not clear whether the implementers intention is that.

Recommendation

We recommend to clarify that with cosmos API as it is not well documented from here https://docs.cosmos.network/v0.39/basics/accounts.html which parameter encrypts the account.

Status: Pending

2. A minor issue

Severity: Minor

https://github.com/Sifchain/sifnode/blob/bebbe9883560bbde4f452f81a2d85bdbc243636a/cmd/ebrelayer/txs/signature.go#L23

The code assumes a threat model whereby the adversary does not have access on the machine of the validator. As such anyone with root access can fetch the private key.

Recommendation

This can be mitigated with a password base access control scheme whenever someone asks for the private key.

Status: Pending

3. An informational issue

Severity: Informational

In order to follow state of the art the implementers may use SHA3 instead of SHA256 and consider Ed25519 for signing as it provides better resistance in side channel attacks and security in case of RNG flaws.

References:

1. http://ed25519.cr.yp.to/
2. http://safecurves.cr.yp.to/
3. http://ed25519.cr.yp.to/ed25519-20110926.pdf