

# AUDITOR LABS

**TON Foundation**  
**Multisignature Wallet**  
**SMART CONTRACT AUDIT**

10.02.2023

**Made by Auditor Labs**

---

## Table of contents

1. Disclaimer.....	3
2. About the Project and Company .....	4
2.1 Project Overview.....	5
3. Vulnerability & Risk Level .....	6
4. Auditing Strategy and Techniques Applied.....	7
4.1 Methodology .....	7
5. Metrics .....	8
5.1 Tested Contract Files .....	8
5.2 Source Unites in Scope .....	9
6. Scope of Work .....	11
6.1 Findings Overview .....	12
6.2 Manual and Automated Vulnerability Test.....	13
6.2.1 Missing Development Environment.....	13
6.3 Verify Claims .....	14
7. Executive Summary.....	15
8. About the Auditor .....	16

## 1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of TON Foundation. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (31.01.2023)	Layout
0.4 (01.02.2023)	Automated Security Testing Manual Security Testing
0.5 (05.02.2023)	Verify Claims and Test Deployment
0.9 (06.02.2023)	Summary and Recommendation
1.0 (10.02.2023)	Final document

## 2. About the Project and Company

**Company address:**

TON Foundation  
One Central 8th and 9th Floor  
Trade Centre - Trade Centre 2  
Dubai - United Arab Emirates



**Website:** <https://ton.org>

**Twitter:** [https://twitter.com/ton\\_blockchain](https://twitter.com/ton_blockchain)

**GitHub:** <https://github.com/ton-blockchain>

**Telegram:** <https://t.me/toncoin>

**LinkedIn:** <https://www.linkedin.com/company/ton-blockchain>

**Medium:** <https://medium.com/ton-community>

## 2.1 Project Overview

TON, or The Open Network, is a unique community-driven blockchain with a lot to offer. Founded by Telegram, TON was created to onboard billions of users and allow for quick, inexpensive and energy-efficient blockchain transactions. All of its features were developed with everyday users in mind. In addition to offering ultra-fast transactions, tiny fees and easy-to-use apps, The Open Network is known for its flexible architecture and scalability. Toncoin (symbol: TON) is the token native to The Open Network (TON), a decentralized Layer 1 blockchain network. Previously known as Gram, Toncoin can be used to pay transaction fees, settle payments or validate transactions using TON blockchain's proof of stake (PoS) consensus model.

As the native token of its blockchain, Toncoin has a range of uses on the TON network, including as a form of payment in decentralized apps (DApps). It does this using blockchain sharding, which involves using multiple subnetworks, or shards, on the same blockchain to quickly accomplish tasks. Each shard has its own purpose and works to prevent large backlogs of unverified blocks. TON is a proof of stake (PoS) network. Every transaction is validated using Toncoin, which is also used to reward validators. The network also allows nominators to lend their tokens to validators to earn rewards. To lend tokens, a nominator needs to join a pool and stake their assets. Both nominators and validators are managed using smart contracts, adding an additional layer of security.

The TON ecosystem is constantly expanding, making Toncoin more versatile. The most common use of the token is as a form of commission payment for processing transactions on the blockchain. Because of the TON network's sharding feature, the token may also be used for cross-chain transaction fees or as payment for creating new work chains.

Some additional use cases include:

- Payment for decentralized data storage
- Payment for the use of TON Proxy
- Payment for TON DNS
- Payments within DApps on the blockchain
- Validator rewards for maintaining the blockchain
- Voting in TON's on-chain governance program

### 3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

## 4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

### 4.1 Methodology

The 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 Auditor Labs to make sure we understand the size, scope, and functionality of the smart contract.
  - 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 Auditor Labs 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 those test cases.
  - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

## 5. Metrics

The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

### 5.1 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
./multisig-code.fc	8eb9c6573485308956d1fc6089f7943c
./show-signed-by.fif	abf11bbe7c30f4a005048867cbc3c29a
./generate-keypairs.fif	3c77b3ba260fe394c0eb49d919237688
./create-msg.fif	a4f0a4a7bea26d75d98ce93be41f823a
./wallet.addr	3a15f8562100a55f6ba46c3fe3173d78
./create-external-message.fif	4d01fc3ac4e901b1c16c13345bef38cd
./add-signature.fif	67a2d574c6ca89fc24a795cbde18da29
./union-signatures.fif	68fa53e742696a10c303972d94d383c4
./new-key.fif	29051b2af2eabd27b39a875e61e9c8ae
./utils.fif	9da86cc6c14e29510d0b60b9df8005a2
./show-msg.fif	f990458d447c75bb3a58628c4ef98a60
./create-order.fif	6ec16d5ea28dd779d78ff5caab26366d
./stdlib.fc	0c38fc5ef43816b67d3ee53942adbaf9
./scheme.tlb	2b7c68da82b75d81f36d55cc971185ed
./clear-signatures.fif	06639b6baebb4dcdbc9840fcd5622840
./show-order.fif	b6662cc68e278e69d17600e0b960cd2c
./new-multisig.fif	83ed5c3a680bea887f2381939f0366b8
./multisig-code.fif	dcbf3337df1865f8a51d9f3d63e770e0



## 5.2 Source Unites in Scope

Source : <https://github.com/ton-blockchain/multisig-contract>

Branch: master

Latest Commit: ae453b676fe40027b40501426586a960812a5b02

Total : 16 files, 2067 codes, 19 comments, 184 blanks, all 2270 lines

filename	language	code	comment	blank	total
Multisig/contracts/add-signature.fif	Fift	41	0	9	50
Multisig/contracts/clear-signatures.fif	Fift	24	0	8	32
Multisig/contracts/create-external-message.fif	Fift	40	0	11	51
Multisig/contracts/create-msg.fif	Fift	51	0	10	61
Multisig/contracts/create-order.fif	Fift	40	0	10	50
Multisig/contracts/generate-keypairs.fif	Fift	19	0	6	25
Multisig/contracts/multisig-code.fc	FunC	260	8	56	324
Multisig/contracts/multisig-code.fif	Fift	642	0	1	643
Multisig/contracts/new-key.fif	Fift	10	0	4	14
Multisig/contracts/new-multisig.fif	Fift	48	0	11	59

<b>filename</b>	<b>language</b>	<b>code</b>	<b>comment</b>	<b>blank</b>	<b>total</b>
Multisig/contracts/show-msg.fif	Fift	29	0	5	34
Multisig/contracts/show-order.fif	Fift	40	0	5	45
Multisig/contracts/show-signed-by.fif	Fift	15	0	5	20
Multisig/contracts/stdlib.fc	FunC	175	11	23	209
Multisig/contracts/union-signatures.fif	Fift	28	0	8	36
Multisig/contracts/utils.fif	Fift	75	0	12	87

## 6. Scope of Work

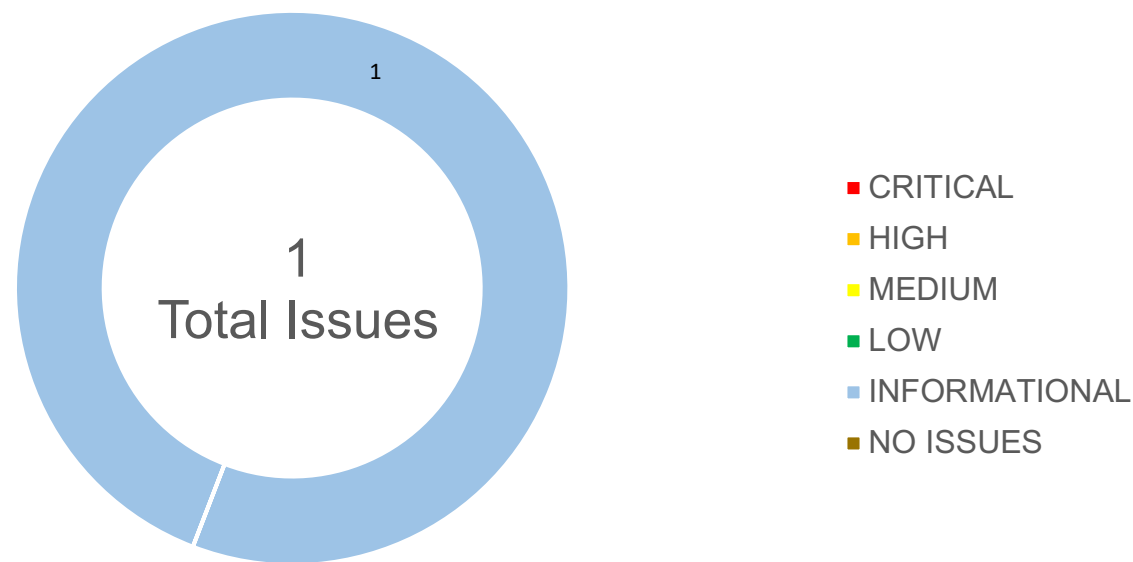
The TON Foundation Team provided us with the files that needs to be tested. The scope of the audit are the Multisignature contracts.

The team put forward the following assumptions regarding the security, usage of the contracts:

- Developer can generate one or several keys with new-key.fif script manually, or generate a batch or keys with generate-keypairs.fif.
- Developer can create an init message for a new multisig wallet with new-multisig.fif script.
- Developer can create one or several internal messages to send with create-msg.fif script.
- Having message(s) to send developer can group it to a new order with create-order.fif script.
- Developer can add a signature to the order with add-signature.fif script
- Having two orders with the same messages to send, but (potentially) different signatures lists, developer can unite the lists with union-signatures.fif script.
- The smart contract is coded according to the newest standards and in a secure way

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.

6.1 Findings Overview



No	Title	Severity	Status
6.2.1	Missing Development Environment	INFORMATIONAL	ACKNOWLEDGED

## 6.2 Manual and Automated Vulnerability Test

### CRITICAL ISSUES

During the audit, experts found **0 Critical issues** in the code of the smart contract.

### HIGH ISSUES

During the audit, experts found **0 High issues** in the code of the smart contract.

### MEDIUM ISSUES

During the audit, experts found **0 Medium issues** in the code of the smart contract.

### LOW ISSUES

During the audit, experts found **0 Low issues** in the code of the smart contract.

### INFORMATIONAL ISSUES

During the audit, experts found **1 Informational issue** in the code of the smart contract.

6.2.1 Missing Development Environment  
Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: NA

File(s) affected: ALL


<b>Attack / Description</b>	The auditors have not recognised an developer friendly environment, including separation of contracts and wrapper or the existence of unit tests.
<b>Code</b>	//
<b>Result/Recommendation</b>	For an easy and error free developer on-ramp, we recommend to turn the project into a TON Blueprint ( <a href="https://github.com/ton-community/blueprint">https://github.com/ton-community/blueprint</a> ) and adding extensive unit tests.

## 6.3 Verify Claims

6.3.1 Developer can generate one or several keys with new-key.fif script manually, or generate a batch or keys with generate-keypairs.fif.

**Status:** tested and verified 


6.3.2 Developer can create an init message for a new multisig wallet with new-multisig.fif script.

**Status:** tested and verified 

6.3.3 Developer can create one or several internal messages to send with create-msg.fif script

**Status:** tested and verified 


6.3.4 Having message(s) to send developer can group it to a new order with create-order.fif script.

**Status:** tested and verified 


6.3.5 Developer can add a signature to the order with add-signature.fif script

**Status:** tested and verified 

6.3.6 Having two orders with the same messages to send, but (potentially) different signatures lists, developer can unite the lists with union-signatures.fif script.

**Status:** tested and verified 

6.3.7 The smart contract is coded according to the newest standards and in a secure way

**Status:** tested and verified 

## 7. Executive Summary

Two (2) independent Auditor Labs experts performed an unbiased and isolated audit of the smart contract codebase.

The main goal of the audit was to verify the claims regarding the security and functions of the smart contract. During the audit, no critical, no high, one medium, no low and one informational issue have been found, after the manual and automated security testing.

We advise the TON Foundation team to implement the recommendation to further enhance the code's security and readability.

## 8. About the Auditor

Auditor Labs is a professional software development firm, founded in 2017 and based in Germany. They show ways, opportunities, risks and offer comprehensive Web3 solutions. Their services include Web3 development, security and consulting.

Auditor Labs conducts code audits on market-leading blockchains such as Solana, Tezos, Ethereum, Binance Smart Chain, and Polygon to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secure the smart contracts of many top DeFi projects.

Auditor Labs currently secures [\\$100 billion](#) in user funds locked in multiple DeFi protocols. The team behind the leading audit firm relies on their robust technical know-how in the web3 sector to deliver top-notch smart contract audit solutions, tailored to the clients' evolving business needs.

Check our website for further information: <https://auditorlabs.com>

### How We Work



**1** -----

#### PREPARATION

Supply our team with audit ready code and additional materials



**2** -----

#### COMMUNICATION

We setup a real-time communication tool of your choice or communicate via e-mails.



**3** -----

#### AUDIT

We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.



**4** -----

#### FIXES

Your development team applies fixes while consulting with our auditors on their safety.



**5** -----

#### REPORT

We check the applied fixes and deliver a full report on all steps done.