PUBLIC

Code Assessment

of the Rates Converter Smart Contracts

August 15, 2025

Produced for

Sky

by

CHAINSECURITY

Contents

1 Executive Summary 3
2 Assessment Overview 5
3 Limitations and use of report 7
4 Terminology 8
5 Open Findings 9
6 Resolved Findings 10
7 Notes 12

1 Executive Summary

Dear all,

Thank you for trusting us to help Sky with this security audit. Our executive summary provides an overview of subjects covered in our audit of the latest reviewed contracts of Rates Converter according to Scope to support you in forming an opinion on their security risks.

Sky implements a rate converter (Conv) to facilitate the conversion between rates per second and yearly rates in basis points (BPS).

The most critical subjects covered in our audit are functional correctness. The general subjects covered are documentation, specifications and gas efficiency.

In summary, we find that the codebase provides a high level of security.

It is important to note that security audits are time-boxed and cannot uncover all vulnerabilities. They complement but don't replace other vital measures to secure a project.

The following sections will give an overview of the system, our methodology, the issues uncovered, and how they have been addressed. We are happy to receive questions and feedback to improve our service.

Sincerely yours,

ChainSecurity

1.1 Overview of the Findings

Below we provide a brief numerical overview of the findings and how they have been addressed.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	1
• Code Corrected	1

2 Assessment Overview

In this section, we briefly describe the overall structure and scope of the engagement, including the code commit which is referenced throughout this report.

2.1 Scope

The assessment was performed on the source code files inside the Rates Converter repository based on the documentation files. The table below indicates the code versions relevant to this report and when they were received.

V	Date	Commit Hash	Note
1	14 Mar 2025	a36dec2706583371bab6a8f338a78025cb6fb1e9	Initial Version
2	20 Mar 2025	a958eb7437b142e95563e3380e2e40e6e673c368	After Intermediate Report
3	23 Jul 2025	26921f3e5aa6bd346d279241c2d1b794ba90c1ae	Fix dataSlot Computation

For the solidity smart contracts, the compiler version `0.8.24` was chosen and the `evm_version` was set to `cancun`.

The following files were in scope:

```
src/Conv.sol
```

2.1.1 Excluded from scope

Generally, all files not mentioned above are out of scope.

2.2 System Overview

This system overview describes the initially received version (Version 1) of the contracts as defined in the Assessment Overview.

Sky offers a rate converter (Conv) to facilitate the conversion between rates per second and yearly rates in basis points (BPS).

Contract Conv is a converter that facilitates the conversion between rates per second and yearly rates in BPS with the following functions. It stores 5001 precomputed per second rates for the corresponding BPS ranging from 0 to 5000, each rate takes 8 bytes.

- `rtob()`: expects an input of yearly rates in BPS (<=5000) and retrieves the corresponding pre-computed rate in RAY (27 decimals).
- `btor()`: expects an input of rates (>=RAY) per second, it computes the yearly rate with auto-compounding first, then converts it to BPS.

2.3 Trust Model

Contract Conv is immutable without any privileged roles.

3 Limitations and use of report

Security assessments cannot uncover all existing vulnerabilities; even an assessment in which no vulnerabilities are found is not a guarantee of a secure system. However, code assessments enable the discovery of vulnerabilities that were overlooked during development and areas where additional security measures are necessary. In most cases, applications are either fully protected against a certain type of attack, or they are completely unprotected against it. Some of the issues may affect the entire application, while some lack protection only in certain areas. This is why we carry out a source code assessment aimed at determining all locations that need to be fixed. Within the customer-determined time frame, ChainSecurity has performed an assessment in order to discover as many vulnerabilities as possible.

The focus of our assessment was limited to the code parts defined in the engagement letter. We assessed whether the project follows the provided specifications. These assessments are based on the provided threat model and trust assumptions. We draw attention to the fact that due to inherent limitations in any software development process and software product, an inherent risk exists that even major failures or malfunctions can remain undetected. Further uncertainties exist in any software product or application used during the development, which itself cannot be free from any error or failures. These preconditions can have an impact on the system's code and/or functions and/or operation. We did not assess the underlying third-party infrastructure which adds further inherent risks as we rely on the correct execution of the included third-party technology stack itself. Report readers should also take into account that over the life cycle of any software, changes to the product itself or to the environment in which it is operated can have an impact leading to operational behaviors other than those initially determined in the business specification.

4 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- *Likelihood* represents the likelihood of a finding to be triggered or exploited in practice
- *Impact* specifies the technical and business-related consequences of a finding
- *Severity* is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.

Likelihood	Impact		
	High	Medium	Low
High	Critical	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

5 Open Findings

In this section, we describe any open findings. Findings that have been resolved have been moved to the Resolved Findings section. The findings are split into these different categories:

- Correctness: Mismatches between specification and implementation

Below we provide a numerical overview of the identified findings, split up by their severity.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	0

6 Resolved Findings

Here, we list findings that have been resolved during the course of the engagement. Their categories are explained in the Open Findings section.

Below we provide a numerical overview of the identified findings, split up by their severity.

Critical-Severity Findings	0
High-Severity Findings	0
Medium-Severity Findings	0
Low-Severity Findings	1

- Incorrect Usage of Keccak Opcode Code Corrected

Informational Findings	1

- Incorrect Specifications Specification Changed

6.1 Incorrect Usage of Keccak Opcode

Correctness Low Version 1 Code Corrected

CS-MK-RTCVT-002

Note: this issue was reported and resolved independently by Sky.

In function `btor()`, when loading the per second rate data, its storage slot must be computed based on the `RATES.slot` and its offset (`wordPos`).

The dataSlot is dependent on `RATES.slot`. When computing the dataSlot, the following computation is used inside of assembly. The `keccak256(p, n)` opcode hashes `n` bytes from memory starting at position `p`. Consequently, it hashes the 32 bytes memory content at position 0 where nothing is stored.

```
let dataSlot := keccak256(RATES.slot, 0x20)
```

This computation has a wrong semantic, however, it works by coincidence since `RATES.slot` is 0.

Code corrected:

Code has been corrected to store the `RATES.slot` into the memory at 0 position and then hashes it.

6.2 Incorrect Specifications

Informational Version 1 Specification Changed

CS-MK-RTCVT-001

The comments of function `btor` suggests the return value is the annual rate value, however, it is actually the per second rate in RAY.

Specification changed:

The NatSpec has been adjusted accordingly.

7 Notes

We leverage this section to highlight further findings that are not necessarily issues. The mentioned topics serve to clarify or support the report, but do not require an immediate modification inside the project. Instead, they should raise awareness in order to improve the overall understanding.

7.1 `rtob` Is Not Restricted to 50%

Note Version 1

Function `btor()` has a restriction for the input yearly rate, which cannot exceed 5000 BPS, since it only has the precomputed results for 0-5000 BPS. However, the function `rtob()` does not have this restriction, hence higher per second rates can be accepted and converted to yearly rates exceeding 5000 BPS.

Users / contracts that interact with Conv should be aware of this to avoid unexpected results.