Bulletproof Plus Rust implementation

Cryptographic security audit

Reference	23-08-1291-LIV
Version	1.1
Date	2023/10/20

Quarkslab SAS
10 boulevard Haussman
75009 Paris
France

Contents

1 Project Information

Document history			
Version	Date	Details	Authors
1.0	2023/10/20	Initial Version	Dahmun Goudarzi
1.1	2023/10/20	Revised version for public release	Dahmun Goudarzi

Quarkslab		
Contact	Role	Contact Address
Frédéric Raynal	CEO	`fraynal@quarkslab.com`
Ramtine Tofighi Shirazi	Project Manager	`mrtofighishirazi@quarkslab.com`
Dahmun Goudarzi	R&D Engineer, Cryptographic Expert	`dgoudarzi@quarkslab.com`

Tari Labs		
Contact	Role	Contact Address
Cayle Sharrock	Head of Engineering	`caylemeister@tari.com`

2 Executive summary

This report presents the results of the collaboration between Tari Labs and Quarkslab on the cryptographic security audit of Tari Labs' Bulletproof Plus [1] implementation in Rust.

Tari is an open-source, decentralized blockchain protocol that is built to enable digital assets to be created, transferred, and managed in a decentralized and privacy-focused way.

To that purpose, Tari Labs developed the Bulletproof Plus protocol in Rust on which this cryptographic security audit is focused.

The Bulletproof Plus implementation is a sensitive part of Tari's project since the protocol is mainly used to prove that a transaction is valid with zero-knowledge techniques, thus providing a privacy protection to end users.

2.1 Purpose

The goal of the cryptographic security audit was to verify the proper implementation of the Bulletproof Plus protocol, the associated cryptographic properties, and verify the optimization applied by Tari Labs. To that end, the following steps were taken:

- **Step 1:** discovery of Tari's code-base and related documentations and specifications with respect to existing articles about Bulletproof Plus [2].
- **Step 2:** review of Tari Labs Bulletproof Plus proper implementation written in Rust with respect to [2] and related specifications.
- **Step 3:** review of cryptographic properties and applied optimization with respect to potential cryptographic attacks and state-of-the-art recommendations.
- **Step 4:** report edition and delivery.

An important effort has been put into the analysis of the specifications in order to write them as depicted in Chapter 6.

2.2 Report structure

This audit report starts with introductory sections and in Section 3 a more detailed presentation of the context and scope of the audit. Then, Section 4 introduces the discovery work performed by Quarkslab auditors to prioritize items for the review and to determine the relevant properties and cryptographic attacks to assess.

Afterward, Section 4.4 presents Quarkslab auditors' review of Tari Labs code-based quality checks. Section 5 details the verifications on the implementation against potential cryptographic attacks and Section 6 provides the assessment of the cryptographic specifications.

Then, Section 7 describes the results of the applied code assessment, using manual review and dynamic testing with the results summarized in Section 8.

Finally, Section 9 provides a conclusion based on the assessment performed on the security of the Tari project Bulletproof Plus implementation, with respect to the above-mentioned scope of work.

The security audit was performed from July to September 2023.

2.3 Disclaimer

This report reflects the work and results obtained within the duration of the audit defined for 30 days on the specified scope and as agreed between Tari Labs and Quarkslab in *23-05-1181-PRO V3.0*. Tests are not guaranteed to be exhaustive and the report does not ensure the code is bug or vulnerability free.

Numerous results in Tari's Bulletproof Plus [**tari_bpp**] are based on academic papers for which we will assume soundness and validity as their review are outside the scope of this audit.

2.4 Findings summary

The following table synthesizes the various findings that were uncovered during the audit. The severity classification given as informative, low, and medium, reflects a relative hierarchy between the various findings of this report. It depends on the threat model and security properties considered.

During the time frame of this assessment, **no critical vulnerabilities were discovered**. The auditors mostly identified low and informative issues as listed above:

- One low issue related to the integration and maintenance of the Merlin crate;
- Two informative issues related to the curve25519 crate and some warnings related to code quality.

The different findings have been disclosed to Tari and addressed by them during the following pull request [3].

ID	Description	Category	Severity
`LOW 1`	The absence of proper maintenance for the Merlin crate is troublesome and it should be considered by Tari to make their own fork to maintain it up to date.	CWE-664: Improper Control of a Resource Through its Lifetime	`Low`
`INFO 1`	Tari's fork for curve25519-dalek is behind upstream	CWE-664: Improper Control of a Resource Through its Lifetime	`Info`
`INFO 2`	Most of the arithmetic operations, if not all, suffer from potential overflow (arithmetic, index, or slice)	CWE-1006: Bad Coding Practices	`Info`

3 Context and scope

This chapter presents the context of the assessment, namely, Tari Labs Bulletproof Plus implementation and optimizations.

In order to get familiar with the context, the auditors introduce next some key concepts.

3.1 Tari

Tari is an open-source, decentralized blockchain protocol that is built on top of the Monero cryptocurrency. It was built to enable digital assets to be created, transferred, and managed in a decentralized and privacy-focused way.

To that purpose, Tari Labs developed the Bulletproofs Plus protocol in Rust on which this cryptographic security audit is focused.

The Bulletproofs Plus implementation is a sensitive part of Tari's project since the protocol is mainly used to prove that a transaction is valid without revealing any information about the inputs or outputs of the transaction, thus providing a privacy protection to end users.

3.1.1 Bulletproofs Plus

Bulletproofs are used in blockchain and cryptocurrency technology to enhance privacy and scalability. They were first introduced in a paper titled "*Bulletproofs: Short Proofs for Confidential Transactions*" by researchers from Stanford University and Blockstream in 2018. Bulletproofs enable efficient and confidential verification of range proofs, which are essential for ensuring the correctness of confidential transactions without revealing the transaction amounts.

Bulletproofs provide several advantages over previous cryptographic techniques like `zk-SNARKs` (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge). They are more efficient in terms of computational requirements and do not require a trusted setup, which is a significant advantage for privacy and security in blockchain applications.

Bulletproofs Plus represents a novel range proof technique, sharing similarities with the foundational Bulletproofs. Both of these range proof protocols are designed to achieve logarithmic proof sizes relative to the number of bits within the range by employing a recursive inner product protocol. However, it's worth noting that while Bulletproofs leverages an enhanced inner product protocol, Bulletproofs Plus distinguishes itself by employing a weighted inner product protocol.

To summarize, Bulletproof Plus compared to Bulletproof provides, among others:

1. **Reduced Size:** One of the primary goals of cryptographic improvements like Bulletproof Plus is to reduce the size of confidential transaction proofs even further compared to Bulletproofs. This reduction in size helps decrease transaction fees and improves overall blockchain scalability.

2. **Improved Efficiency:** Bulletproof Plus aims to enhance the efficiency of the proof generation and verification processes, making it faster and less resource-intensive. This can benefit both users and network validators.
3. **Compatibility:** It's crucial for upgrades like Bulletproof Plus to be backward-compatible with existing transaction formats and protocols to ensure a smooth transition for users and maintain compatibility with older software.

On top of that, Tari Labs make use of a result provided by two anonymous researchers in [4] that allows to extend Bulletproofs Plus proofs to accommodate multiple masks.

Finally, the auditors note that some optimizations were directly applied and implemented by Tari Labs on their Bulletproof Plus Rust implementation [**tari_bpp**] and for which documentations were provided through internal RFC [5].

3.2 Scope and methodology

As previously mentioned, the purpose of this cryptographic security review is to verify the proper implementation of the Bulletproofs Plus protocol, the associated cryptographic properties, and verify the optimization applied by Tari Labs.

The processes applied for the security review of Tari's Bulletproofs Plus implementation was as follows:

1. Deep-dive into the general cryptographic principles behind the algorithms and the different optimizations.
2. Reconstruction and analysis of the overall specification of the Tari's BulletproofsPlus implementation.
3. Assessment of the code quality and implementation with respect to the specification and of the optimizations with the security requisites.
4. Assessing arithmetic operations conformity, namely in the fork of the `curve25519-dalek` crate.
5. Fuzzing for arithmetic overflows.
6. Fuzzing for breach of cryptographic security goals.

4 Discovery

The first step of the cryptographic security audit was focused on the project code base, documentations, and specifications discovery. The purpose of the discovery phase is to gain a comprehensive understanding of the cryptographic system under evaluation and sets the stage for the more in-depth analysis and testing phases.

It helps auditors build a comprehensive picture of the system, its design, and its potential security risks. To that end, this chapter briefly presents

1. The documentations review performed to understand the intended functionalities, security requirements, and design principles.
2. The codebase discovery to gain knowledge towards its quality assessment (see Section 4.4) and security review (see Chapter 7).
3. The dependency identification to check for proper maintenance and potentially known vulnerabilities that may impact the audited project.

4.1 Bulletproofs Plus State of the Art

In the section, we describe the different documentations and library that have been studied by the auditors to build some knowledge base around Tari's implementation of Bulletproofs Plus. We first start by listing different resources for understanding better what are range proofs, Bulletproof, and the main differences between Bulletproof and Bulletproofs Plus. Then, we list the different papers and internal documents used by Tari in order to produce a Rust Bulletproofs Plus implementation.

4.1.1 Range Proof and Bulletproof

In order to familiarize ourselves with range proofs and Bulletproof, we scheme through different resources available. To have a good understanding on the design, the rationale, and the application of Bulletproof the video from one of the original author at BPASE 2018 [6] is a good starting point. The documentation of the `Dalek` crate which is used by Tari Labs for the implementation of BulletproofsPlus provides some good insides on how range proofs work and more specifically for Bulletproof [7], as well as a blog post made by one of the main contributor of the Rust implementation of Bulletproof [8].

4.1.2 Bulletproofs Plus

Bulletproofs Plus was introduced by Chung *et al.* in a preprint article in 2020 [2]. There exists a nice blog post explaining the main differences between Bulletproof and Bulletproofs Plus and efficiency comparison by Suyash Bagad [9].

4.1.3 Tari's BulletproofsPlus

The two major changes in Tari Labs implementation of Bulletproofs Plus come from both the appendix of a preprint written by two anonymous authors [4] and Tari's internal RFC documentation [5].

4.2 Code Structure

The Bulletproof Plus Rust implementation is composed of 3 main directories:

1. `src`,
2. `benchmark`,
3. `tests`.

The other directories and files reported by the `tree` command are considered miscellaneous (versioning scripts, code quality, etc.).

```
> tree -I target
.
|-- CHANGELOG.md
|-- Cargo.lock
|-- Cargo.toml
|-- LICENSE
|-- README.md
|-- benches
|   |-- generators.rs
|   |-- range_proof.rs
|-- lints.toml
|-- rust-toolchain.toml
|-- rustfmt.toml
|-- scripts
|   |-- cargo-version-updater.js
|   |-- file_license_check.sh
|-- src
|   |-- commitment_opening.rs
|   |-- errors.rs
|   |-- extended_mask.rs
|   |-- generators
|   |   |-- aggregated_gens_iter.rs
|   |   |-- bulletproof_gens.rs
|   |   |-- generators_chain.rs
|   |   |-- mod.rs
|   |   |-- pedersen_gens.rs
|   |-- inner_product_round.rs
|   |-- lib.rs
|   |-- protocols
|   |   |-- curve_point_protocol.rs
|   |   |-- mod.rs
|   |   |-- scalar_protocol.rs
|   |   |-- transcript_protocol.rs
|   |-- range_parameters.rs
|   |-- range_proof.rs
|   |-- range_statement.rs
|   |-- range_witness.rs
|   |-- ristretto.rs
|   |-- traits.rs
|   |-- transcripts.rs
|   |-- utils
|       |-- generic.rs
|       |-- mod.rs
|       |-- non_debug.rs
|-- test_coverage.sh
|-- tests
    |-- ristretto.rs

8 directories, 38 files
```

4.2.1 src/

The `src/` repository is the core of the Bulletproofs Plus Rust implementation. This is where the two main functions for inner products and range proofs are defined. This is where the generators, the protocols (based on Merlin transcripts), the cryptographic utilities, and the implementation for Ristretto for `Curve25519` are defined. The key files are `src/inner_product_round.rs`, `src/range_proof.rs`, which implements the two main protocols.

4.2.2 benchmark/

The benchmark repository is composed of benchmarking of range proof functions (create_rp, verify_rp, verify_batched_rp) for different n: n_{small} and n_{64}.

The full spectrum of benchmarks are commented and not by default, most likely for efficiency reasons as the full spectrum can be quite extensive.

4.2.3 tests/

The tests repository is only composed of one file called `ristretto.rs`. That can be a bit counter-intuitive since it only represents 4 tests out of the around 30 present in the projects. Most of the tests are done inside the different src files and not in this repository.

The 4 tests done on different types of proofs:

1. non-aggregated single proof multiple bit lengths,
2. non-aggregated multiple proofs single bit lengths,
3. aggregated single proof multiple bit lengths,
4. and mixed aggregation multiple proofs single bit length.

4.3 Dependencies

4.3.1 Merlin's Transcript

Merlin is a Rust crate authored by Henry de Valence [10] to construct transcripts, for instance to be used for zero-knowledge proofs, by automating the Fiat-Shamir transform based on STROBE. One of the main advantages in using Merlin is the domain separators that ensure challenges to be bound to the statements to be proved, which when properly used allows to avoid replay attacks (see Section 5). The last version is from two years ago and does not seem to be maintained.

`LOW 1`	Merlin's crate is of outmost importance for the security of Tari's implementation. The absence of maintenance is troublesome and it should be considered by Tari to make their own fork to maintain it up to date.	
Category	CWE-664: Improper Control of a Resource Through its Lifetime	
Rating	**Impact**: Supply Chain	**Exploitability**: None

4.3.2 `tari-curve25519-dalek`

Tari's Bulletproofs Plus implementation uses their own fork of the `curve25519-dalek` crates [11], named `tari-curve25519-dalek`. The current version is v4.0.3, which was not updated since July 12th. The fork is not up to date with latest version of the `curve25519-dalek` crate.

We ran a *diff* between the two versions using the *Meld* diffing tool. The changes are minor and are there to adapt some of the codes to the specifically tailored case of Tari. However, we noticed that the latest version of Dalek has a different implementation of some arithmetic operations such as in the Montgomery scalar multiplication where an other algorithm is used (Algorithm 8 of Costello-Smith 2017).

The fork version from Tari should be kept up to date, specially if security patches are applied on the upstream repository.

`INFO 1`	Tari's fork for curve25519-dalek is behind upstream	
Category	CWE-664: Improper Control of a Resource Through its Lifetime	
Rating	**Impact**: Supply Chain	**Exploitability**: None

4.4 Code quality

One of the first things we like to do when we encounter a Rust project is to have some ideas about the code quality. Code quality is an important aspect of the review for the following reasons:

- It provides an overview of the project implementation in terms of memory and concurrency safety and potential errors that could lead to bugs and/or potential vulnerabilities.
- It assesses the code maintainability and auditability which is important in terms of security practices, especially for open-source projects, and also helps auditors determine the next steps of the security assessment.
- It evaluates the project's dependencies management, which is crucial for ensuring that the code base is not vulnerable via a known security issue in third-party libraries.

Overall, a well-maintained and high-quality project provides community trust and can lead to valuable code reviews, contributions, and security audits.

To that end, several tools from the Rust ecosystem are useful to assess the project's code quality, as presented next.

4.4.1 cargo audit

Running `cargo audit` gives some hints on the state of the dependencies.

```
qb@tari-bpp:~/bulletproofs-plus > cargo audit
        Fetching advisory database from
        ↪  `https://github.com/RustSec/advisory-db.git`
          Loaded 555 security advisories (from /Users/qb/.cargo/advisory-db)
        Updating crates.io index
        Scanning Cargo.lock for vulnerabilities (116 crate dependencies)
```

We can see here than this project use no external dependencies that have known issues.

External dependencies management shows no issues which is a good point.

4.4.2 cargo geiger

Incorrect use of unsafe code can lead to several issues (e.g., memory safety, undefined behaviors) which are some of the problems Rust aims to prevent in safe code. Therefore, unsafe code should be avoided when possible or used very carefully in addition to thorough testing and documentations to ensure correctness.

`cargo geiger` checks if dependencies use unsafe code. The output is quite long and can be checked in Appendix A.1. We can notice that several dependencies are using unsafe code. However, this does not mean that the code is inherently unsafe.

Type	Occurrence	Location
Arithmetic	3	22:13, 29:13, 35:20
Index	2	30:19, 30:19
Slicing	0	-

Table 4.3: src/generators/aggregated_gens_iter.rs: 5 warnings

Type	Occurrence	Location
Arithmetic	0	-
Index	2	75:13, 78:13
Slicing	0	-

Table 4.4: src/generators/bulletproof_gens.rs: 2 warnings

It is to be noted that the bulletproofs-plus project does not use the forbid unsafe attribute, which is a rule that forbids unsafe code in the current crate. However, we can see that the crate does not use any unsafe code.

Project's crate does not use any unsafe code which is a good point.

4.4.3 cargo clippy

`cargo clippy` is a linter to catch common mistakes and improve code quality in Rust. The output is quite long as we caught 220 warnings using the following arguments that can be checked in Appendix A.2. In order to have a better understanding of the numerous warnings caught by `cargo clippy`, we regrouped them into their respective described category. We noticed that only 3 types of warnings occur, the three of them being related to arithmetic operations, which are the following ones:

- 110 occurrences: *arithmetic operation that can potentially result in unexpected side-effects*[1],
- 95 occurrences: *indexing may panic*[2],
- 22 occurrences: *slicing may panic*[3].

More precisely, around 90% of the warnings (200 out of 220) are occurring in only 2 files: `src/inner_product_round.rs` (60 warnings) and `src/range_proof.rs` (140 warnings). This is not surprising as the main cryptographic operations related to Bulletproofs Plus are made in those two files.

To summarize, the findings are categorized as informative since the client was made aware of them in early stages of the audit and started working on the fixes. We made tables for each files on the number and types of warnings caught by `cargo clippy` and where to find them, and provided below.

[1]https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects
[2]https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing
[3]https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

Type	Occurrence	Location
Arithmetic	22	86 : 90, 111 : 36, 112 : 36, 146 : 26 149 : 25, 151 : 17, 152 : 17, 160 : 28 161 : 28, 162 : 28, 164 : 30, 186 : 43 187 : 43, 205 : 9, 210 : 13, 211 : 13 238 : 22, 239 : 22, 245 : 55, 260 : 24 261 : 32, 263 : 13
Index	30	146 : 27, 147 : 18, 148 : 37, 148 : 56 148 : 73, 148 : 92, 149 : 45, 151 : 24 151 : 41, 152 : 23, 152 : 40, 160 : 32 161 : 32, 164 : 30, 164 : 39, 164 : 50 179 : 30, 184 : 13, 187 : 47, 210 : 20 210 : 28, 210 : 51, 211 : 20, 211 : 28 211 : 55, 238 : 14, 239 : 14, 263 : 30 263 : 50, 263 : 13
Slicing	8	171 : 19, 172 : 19, 173 : 19, 174 : 19 175 : 27, 176 : 27, 177 : 27, 178 : 27

Table 4.5: src/inner_product_round.rs: 60 warnings

Type	Occurrence	Location
Arithmetic	2	66:22, 79:22
Index	5	66:23, 66:13, 79:23, 79:31, 79:13
Slicing	2	52:9, 52:44

Table 4.6: src/protocols/curve_point_protocol.rs: 9 warnings

Type	Occurrence	Location
Arithmetic	2	54:22, 65:22
Index	5	54:22, 54:13, 65:22, 65:29, 65:13
Slicing	0	-

Table 4.7: src/protocols/scalar_protocol.rs: 7 warnings

Type	Occurrence	Location
Arithmetic	80	234 : 43, 235 : 43, 243 : 43, 250 : 27 273 : 24, 276 : 47, 278 : 21, 279 : 27 283 : 40, 287 : 20, 291 : 24, 297 : 13 300 : 13, 305 : 13, 307 : 17, 361 : 26 389 : 16, 390 : 26, 400 : 29, 489 : 13 496 : 31, 498 : 9, 510 : 35, 512 : 13 546 : 33, 552 : 30, 585 : 28, 586 : 28 592 : 24, 593 : 36, 594 : 40, 596 : 26 599 : 25, 600 : 17, 601 : 17, 605 : 44 608 : 24, 612 : 28, 619 : 37, 621 : 25 622 : 32, 625 : 13, 634 : 49, 638 : 29 640 : 33, 641 : 33, 643 : 29, 665 : 29 667 : 24, 670 : 25, 671 : 25, 672 : 17 673 : 17, 674 : 17, 675 : 17, 681 : 17 682 : 32, 685 : 21, 690 : 13, 692 : 17 695 : 34, 697 : 34, 699 : 34, 702 : 70 704 : 74, 815 : 42, 836 : 32, 842 : 28 843 : 27, 848 : 43, 859 : 74, 861 : 21 862 : 74, 865 : 19, 867 : 71, 868 : 70 869 : 81, 871 : 81, 877 : 28, 900 : 32
Index	50	237 : 59, 279 : 27, 287 : 26, 291 : 24, 300 : 22 300 : 29, 307 : 48, 307 : 48, 307 : 75, 359 : 37 359 : 73, 360 : 26, 361 : 26, 361 : 60, 363 : 32 365 : 63, 385 : 68, 395 : 13, 396 : 13, 411 : 21 419 : 21, 478 : 37, 478 : 73, 479 : 26, 480 : 32 481 : 34, 482 : 33, 483 : 23, 512 : 75, 577 : 86 577 : 103, 593 : 36, 593 : 52, 594 : 40, 594 : 60 608 : 30, 612 : 28, 640 : 46, 641 : 46, 667 : 24 667 : 35, 670 : 44, 671 : 34, 672 : 17, 673 : 65 673 : 17, 692 : 47, 692 : 17, 713 : 34, 714 : 33
Slicing	10	841 : 71, 859 : 68, 862 : 68, 866 : 64, 867 : 65 868 : 64, 869 : 75, 871 : 75, 877 : 22, 905 : 48

Table 4.8: src/range_proof.rs: 140 warnings

Type	Occurrence	Location
Arithmetic	0	-
Index	1	27:32
Slicing	0	-

Table 4.9: src/range_witness.rs: 1 warning

Type	Occurrence	Location
Arithmetic	1	74:17
Index	0	93:32, 100:31
Slicing	2	-

Table 4.10: src/utils/generic.rs: 3 warnings

INFO 2	Numerous arithmetic operations suffer from potential overflow (arithmetic, index, or slice)	
Category	CWE-1006: Bad Coding Practices	
Rating	**Impact**: Code quality	**Exploitability**: None

5 Evaluation Overview

5.1 Hypothesis

The following hypothesis has been taken into account in order to evaluate Tari's Bulletproofs Plus implementation.

Sound Protocols

The underlying protocols coming from both papers, BulletproofsPlus [2] and Zarcanum [4] are sound: we do not evaluate the paper cryptographic content. Please note that for BulletproofsPlus, an audit was made where the security proofs of BulletproofsPlus were backed up [12].

Sound Merlin crate

The Merlin crate is sound: we do not evaluate the external crate that allows to use Fiat-Shamir transform on the BulletproofsPlus protocol but rather how it is used.

Side-channel attacks out-of-scope

We are not considering side-channel attacks on the device manipulating secret values (namely, the prover's device).

5.2 Threats and Security checks

There are three main security paradigms in Bulletproofs Plus, which are the followings:

Perfect completeness

which means that an honest prover will always convince an honest verifier on a true statement.

Perfect Honest Verifier Zero-Knowledge

which means transcripts can be perfectly simulated and leak no information whatsoever.

Computational Witness-Extended Emulation

which means that there exists an emulator (that's allowed to rewind the prover to any state, and to supply the verifier with fresh randomness) that extracts a witness from the (modified) transcript.
For simplicity, it is a generalization of the notion of perfect soundness where it is impossible to prove a false statement.

One of the main threats in interactive protocols such as Bulletproofs Plus is **replay attacks and transferability**: ones need to avoid that a proof is copied and used in a different context by the verifier. It is therefore of most importance to verify that the transcripts are properly formed and used in Tari's BulletproofsPlus implementation.

5.3 Methodology

The processes applied for the security review of Tari's Bulletproofs Plus implementation were as follows:

- Deep-dive into the general cryptographic principles behind the algorithms and the different optimizations.
- Reconstruction and analysis of the overall specification of the Tari's BulletproofsPlus implementation.
- Assessment of the implementation with respect to the specification and of the optimizations with the security requisites.
- Assessing arithmetic operations conformity, namely in the fork of the `curve25519-dalek` crate.
- Assessing nonce generation behavior.
- Fuzzing for arithmetic overflows.
- Fuzzing for breach of cryptographic security goals.

The main objective was to uncover potential cryptographic vulnerabilities and/or bugs and to assess the resiliency of the project against the above-mentioned cryptographic threats.

5.4 Topics covered

After a proper discovery of the code base as detailed in Section 4, a priority assessment has been established on the severity of each part of the code.

5.4.1 Range Proof and Inner Product Round

These are the two main functions implementing the Bulletproofs Plus protocols. Any mistakes or non-compliance with the specification will most likely infer a loss in one of the three main security properties.

Their implementations can be found in `src/range_proof.rs` and `inner_product_round.rs` respectively.

5.4.2 Transcripts

The functions allowing the protocol to be non-interactive and upon which most of the security against replay attacks relies. The implementation of those functions relies on the use of the Merlin's crate.

Definition of transcript functions can be found in `src/transcript.rs`, and are called and used in different parts of the code (mainly in the two protocol functions aforementioned).

5.4.3 Generators

Different types of generators are needed in range proofs of the Bulletproof type to avoid the need of a trusted setup. The proper generation and definition of such generators have an impact in the security of the protocol.

Their implementations can be found in the `generator` subfolder which defines structures containing all the generators needed for aggregating up to m range proofs of up to n bits each.

5.4.4 Curve25519 Operations

The main arithmetic cryptographic operations are curve operations such as scalar multiplication and multi-scale exponentiations which are defined in an external crate forked and modified by Tari's developers. Improper implementation of such operations can lead to vulnerabilities.

The API functions with the `tari-curve25519-dalek` are defined in `protocols/curve_point_protocols.rs`.

5.5 Protocol Specification

During the discovery and the code review, we noticed that no overall specifications of the protocols implementation were provided. Namely the Rust implementation is based on a mash-up of two academic papers, different internal optimizations, an external crate for the Fiat-Shamir transformation, a home-made fork of the Dalek crate which performs the arithmetic cryptographic operations. The internal documentation in `RFC-0181` only described a portion of the optimization made by Tari themselves, since we found out that some of them were directly in comments of the code or in undocumented changes in the Dalek crate.

While we asked for a proper specification in early stages of the audit and we were told that Tari did not want to reproduce what was directly taken from papers, we do believe after carefully doing the evaluation that not having access to a proper specification of the overall protocols is a step backwards to a proper security audit. Discovering new design decisions during code review should be prohibited and can be seen as something orthogonal to Kerckhoffs' principle, which is a fundamental concept in cryptography.

Moreover, a proper specification will allow the code to be clearer and to have unified variable notations, which sometimes made the code review troublesome.

> We recommend to write a proper specification of the implementation that involves all design choices.

In the following section, we tried to reproduce a comprehensive specification of the overall implementations based on the two academic papers Bulletproofs Plus, Zarcanum, and Tari's internal RFC. The provided specification was also assessed as part of this cryptographic security audit.

6 Protocol Specification Analysis

The security analysis of the cryptographic protocol specifications is a crucial step for the security auditors in order to:

- Understand the modification with respect to previous work and scientific publications on the Bulletproof and Bulletproof Plus [2] protocols.
- Ensure that the cryptographic algorithms and protocols are designed with security in mind.
- Assess the specifications to identify potential vulnerabilities and evaluate the attack surface.
- Finally, to evaluate the resiliency against mentioned threats (see Section 5.2).

As explained in Section 4, the Bulletproofs Plus Rust implementation is based on the Bulletproofs Plus paper [2], the Zarcanum paper [4], and some in-house optimizations [5]. In this section, we provide a detailed specification and associated security analysis of the final protocol, when all these results are combined together.

Please note that on the following figures, we described the case of m aggregated proof, and the non-aggregated case is simply the case where $m = 1$.

The different colors illustrated in the following figures are used to specify the different features added to the original Bulletproofs Plus:

- blue: Zarcanum's commitment extensions,
- orange: Tari's global changes and optimizations,
- red: Tari's specific use-case tailored optimizations.
- teal: Optimization informally described in BulletproofPlus [2].

In the following, we provide a more in-depth explanation of these features.

6.1 Notation

Depicted in orange in our specifications, they only concern the Figure 6.2 where the variable A was replaced with A' to avoid confusion with the variable A from the range proof. We decided to discard the change of notation for the basis from [5] in this description as they are only adding confusion to the comprehension of the different papers [2, 4] and did not seem to be taken into account in the implementation (more details in Section 7).

6.2 Commitments Extensions

In the appendixes of a preprint [4], two anonymous authors proposed a way to extend commitments to values using multiple masks.

Relation:

$$\{\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, g, h_1, \ldots, h_m, P \in \mathbb{G}, \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n, \alpha_1, \ldots, \alpha_m \in \mathbb{Z}_p :$$
$$P = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} g^{\mathbf{a} \odot_y \mathbf{b}} h_1^{\alpha_1}, \cdots, h_m^{\alpha_m}\}$$

Prover
Input: $(\mathbf{g}, \mathbf{h}, g, h_i, \mathbf{V}, \mathbf{v}, \gamma_i)$
Output: $\varnothing$

Verifier
Input: $(\mathbf{g}, \mathbf{h}, g, h_i, \mathbf{V})$
Output: Accept or Reject

For $j \in [1, m]$,
Let $\mathbf{d}_j = (0, \ldots, 0, 2^n, 0, \ldots, 0)$,
Chooses $\alpha_i \leftarrow\$ (\mathbb{Z}_p$ or *shared_seed*$)$
Set $\mathbf{a}_L \in [0,1]^{m\dot{n}}$s.t $\langle \mathbf{a}_L, \mathbf{d}_j \rangle = v_j - v_{min}$,
$\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^{m\dot{n}} \in \mathbb{Z}_p^{m\dot{n}}$
Compute $A = \mathbf{g}^{\mathbf{a}_L}\ \mathbf{h}^{\mathbf{a}_R}\ h_i^{\alpha_i}$

$$\xrightarrow{A,\, v_{min}}$$

$y \leftarrow\$ \mathbb{Z}_p$, $z \leftarrow\$ \mathbb{Z}_p$

$$\xleftarrow{y,\, z}$$

Let $\mathbf{d} = \sum_{j=1}^{m} z^{2j} \cdot \mathbf{d}_j$

$$\text{Compute } \hat{A} = A \cdot \mathbf{g}^{-\mathbf{1}^{mn} \cdot z} \mathbf{h}^{\mathbf{d} \circ \overleftarrow{y}^{mn} + \mathbf{1}^{mn} \cdot z} \cdot (\prod_{j=1}^{m} (V_j - v_{min} \cdot g)^{z^{2j}})^{y^{mn+1}}$$
$$\cdot\, g^{\langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z - (2^n - 1) \sum_{j=0}^{m-1} z^{2(j+1)} \cdot z y^{mn+1} - \langle \mathbf{1}^{mn}, \overrightarrow{y}^{mn} \rangle \cdot z^2} \in \mathbb{G}$$

Compute:
$\hat{\mathbf{a}_L} = \mathbf{a}_L - \langle \mathbf{1}^{mn}, z \rangle \in \mathbb{Z}_p^{mn}$
$\hat{\mathbf{a}_R} = \mathbf{a}_R + \mathbf{d} \circ \overleftarrow{y}^{mn} + \mathbf{1}^{mn} \cdot z \in \mathbb{Z}_p^{mn}$
$\hat{\alpha}_i = \alpha_i + \sum_{j=1}^{m} z^{2j} \cdot \gamma_{ji} \cdot y^{mn+1} \in \mathbb{Z}_p$

Run $\texttt{ZK-WIP}_{\overrightarrow{y}^{mn}}(\mathbf{g}, \mathbf{h}, g, h_i, \hat{A}, \hat{\mathbf{a}_L}, \hat{\mathbf{a}_R}, \hat{\alpha}_i)$

Figure 6.1: Tari's Zero-Knowledge Argument for Range proof $v_j \in [0, 2^n - 1]$ for $j \in [1, m]$.

Prover

Input: $(\mathbf{g}, \mathbf{h}, g, h_i, P, \mathbf{a}, \mathbf{b}, \alpha_i)$

Output: $\varnothing$

Verifier

Input: $(\mathbf{g}, \mathbf{h}, g, h_i, P)$

Output: Accept or Reject

$\boxed{n = 1}$

Let $r, s, \delta_i, \eta_i \hookleftarrow\$ (\mathbb{Z}_p$ or $\mathit{shared\_seed})$

Compute $A' = \mathbf{g}^r \mathbf{h}^s g^{r \odot_y \mathbf{b} + s \odot_y \mathbf{a}} h_i^{\delta_i} \in \mathbb{G}$,

$B = g^{r \odot_y s} h_i^{\eta_i} \in \mathbb{G}$

$\xrightarrow{A', B}$

$e \hookleftarrow\$ \mathbb{Z}_p^*$

$\xleftarrow{e}$

Compute $r' = r + \mathbf{a} \cdot e \in \mathbb{Z}_p$

$s' = s + \mathbf{b} \cdot e \in \mathbb{Z}_p$

$\delta'_i = \eta_i + \delta_i \cdot e + \alpha \cdot e^2 \in \mathbb{Z}_p$

$\xrightarrow{r', s', \delta'_i}$

Accept if:

$$(\prod_{j=1}^{m} (V_j - v_{min} \cdot g)^{z^{2j}})^{y^{mn+1} e^2} A^{e^2} \cdot \Big(\prod_{j=1}^{\log_2(mn)} L_j^{e^2 \cdot e_j^2} R_j^{e^2 \cdot e_j^{-2}} \Big) A'^e B =$$
$$\mathbf{g}^{r' e \cdot \boldsymbol{\sigma} + e^2 (z \cdot \mathbf{1}^{mn})} \mathbf{h}^{s' e \cdot \boldsymbol{\sigma}' - e^2 (z + \mathbf{d} \circ \overleftarrow{y}^{mn})}$$
$$g^{r' \odot_y s' - e^2 x + e^2 y^{mn+1} \sum_{j=0}^{m-1} z^{2(j+1)} v_{min,j}} h_i^{\delta'_i}$$

Reject otherwise

$\boxed{n > 1}$

Let $\hat{n} = \frac{n}{2}, \mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2), \mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2)$,

$\mathbf{g} = (\mathbf{g}_1, \mathbf{g}_2)$, and $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2)$.

Let $d_{Li}, d_{Ri} \hookleftarrow\$ (\mathbb{Z}_p$ or $\mathit{shared\_seed})$

Compute $c_L = \mathbf{a}_1 \odot_y \mathbf{b}_2 \in \mathbb{Z}_p$

$c_R = (y^{\hat{n}} \mathbf{a}_2) \odot_y \mathbf{b}_1 \in \mathbb{Z}_p$

$L = \mathbf{g}_2^{y^{-\hat{n}} \cdot \mathbf{a}_1} \mathbf{h}_1^{\mathbf{b}_2} g^{c_L} h^{d_{Li}} \in \mathbb{G}$

$R = \mathbf{g}_1^{y^{\hat{n}} \cdot \mathbf{a}_2} \mathbf{h}_2^{\mathbf{b}_1} g^{c_R} h^{d_{Ri}} \in \mathbb{G}$

$\xrightarrow{L, R}$

$e \hookleftarrow\$ \mathbb{Z}_p^*$

$\xleftarrow{e}$

Compute $\hat{\mathbf{g}} = \mathbf{g}_1^{e^{-1}} \circ \mathbf{g}_2^{e \cdot y^{-\hat{n}}} \in \mathbb{G}^{\hat{n}}$

$\hat{\mathbf{h}} = \mathbf{h}_1^{e} \circ \mathbf{h}_2^{e^{-1}} \in \mathbb{G}^{\hat{n}}$

$\hat{P} = L^{e^2} P R^{e^{-2}} \in \mathbb{G}$

Compute $\hat{\mathbf{a}} = \mathbf{a} \cdot e + (\mathbf{a}_2 \cdot {}^{y^{\hat{n}}}) \cdot e^{-1} \in \mathbb{Z}_p^{\hat{n}}$

$\hat{\mathbf{b}} = \mathbf{b} \cdot e^{-1} + \mathbf{b}_2 \cdot e \in \mathbb{Z}_p^{\hat{n}}$

$\hat{\alpha} = d_{Li} \cdot e^2 + \alpha + d_{Ri} \cdot e^{-2} \in \mathbb{Z}_p$

Run $\texttt{ZK-WIP}_{\vec{y}^{\hat{n}}}(\hat{\mathbf{g}}, \hat{\mathbf{h}}, g, h, \hat{P}, \hat{\mathbf{a}}, \hat{\mathbf{b}}, \hat{\alpha})$

Figure 6.2: Tari's Zero-Knowledge Argument for WIP relation.

This result is done first by showing that the BulletproofsPlus [2] protocol can be modified in order to use one additional mask, whose proof followed the same rationale as in the Bulletproofs Plus article proof. Then, by induction, they can extend the result to multiple masks. This only implies slight modification of the BulletproofsPlus protocol (namely adding some indexes to the corresponding values) and are depicted in blue in Figure 6.1 and 6.2.

6.3 Minimum Value Assertion

Depicted in orange in our specifications, this optimization proposed by Tari in RFC-0181 [5] is a change of variable in the prover's commitment in order to insert a lower bound different than 0 and change the upper bound allowed for value blinding. This lower bound, v_{min} is specified by the prover, which then uses $v - v_{min}$ as the value witnesses and transmits the lower bound to the verifier. The verifier can verify the proof by using $V - v_{min}G$ as the commitment.

6.4 Batch Verification

In order to preserve precious CPU time, Tari uses the batch verification techniques informally described in BulletproofsPlus [2], where each proof verification function gets a corresponding random multiplicative scalar weight. This allows to combine the verification of each proof into a single multi-scalar multiplication.

This technique is depicted in teal in our specification. To avoid confusion with the already existing notation s and s', we decided in this report to rename the variables s and s', $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma'}$, namely we have that $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$, $\boldsymbol{\sigma'} = (\sigma'_1, \ldots, \sigma'_n) \in \mathbb{Z}_p^n$ where

$$\sigma_i = (\frac{1}{y^{i-1}}) \cdot \prod_{j=1}^{\log_2(n)} e_j^{b(i,j)} \text{ and } \sigma'_i = \prod_{j=1}^{\log_2(n)} e_j^{-b(i,j)}.$$

For batch verifications, we can then add the random weights to the teal equation from Figure 6.2 as follows:

$$\sum_{l=1}^{\#proofs} w_l \cdot Proof_i \stackrel{?}{=} \mathbf{0}.$$

Where $Proof_i$ corresponds to the right-hand side of the teal equation minus the left-hand side.

6.5 Mask Recovery

For the case of non-aggregated range proof for a range assertion of a single commitment, Tari introduced in RFC-0181 [5] a way to sample the different nonces of the proof via a shared nonce seed. This allows for the verification to recover the masks γ_i as follows:

$$\gamma_i = \Big((\delta'_i - \eta_i - \delta_i e)e^{-2} - \alpha_i - \sum_{j=0}^{\log(n)-1} (e_i^2 d_{L,j,i} + e_j^2 d_{R,j,i}) y^{-(n+1)} z^{-2}\Big).$$

Sampling for the shared nonce seed when doing a non-aggregated range proof for a range assertion of a single commitment is depicted in red.

6.6 Sum Optimization

This optimization proposed by Tari in RFC-0181 [5] is based on mathematical results from geometric series which allows to speed up some computations. This optimization is depicted in orange as well and consists in replacing sums of d_i as follows:

$$\langle \mathbf{1}^{mn}, \mathbf{d} \rangle = (2^n - 1) \sum_{j=0}^{m-1} z^{2(j+1)}$$

7 Code Review

This section presents the results of the code review performed on Tari's Bulletproofs Plus Rust implementation. The purpose of the code review was mainly to assess the implementation with respect to the specification as detailed in Chapter 6 as well as the resiliency of the implementation against the cryptographic threats that may apply, as detailed in Section 5.

Overall the code is well written and is for the most part consistent with the specifications. When necessary, functions have unit tests implemented.

The unit tests mostly cover edge cases of the code and could also verify that the corresponding code is doing the correct functionalities.

We noticed that some implementations decisions on key elements that can have significant impact on the security are detailed in comments of the code but not in Tari's internal documentations, which can create a delta between what is implemented and the specification.

The auditors suggest Tari to detail security-related key elements in their documentations in addition to what is already detailed in the code's comments.

Some notations are sometimes confusing by either not following the specifications or the papers upon which the protocol is based (for instance the different generators) which can make the comprehension and the reading of the code quite cumbersome. Moreover, naming of variable should follow the same convention throughout the code. For instance, curve Point is usually denoted with upper-case letters in cryptographic codes and in this implementation but we noticed several cases where they were in lower cases.

In the followings, we go into detail on some portions of the code that raised attention during code review.

Please note that even though files do not appear in the following, all the source material involved in the considerations from Section 5 was carefully reviewed.

7.1 Basis notations

There are three notations for the different basis involved in the Bulletproof Plus protocol between the academic papers, Tari's internal RFC, and the implementation. This has made the review quite cumbersome and require more assessment time. Moreover, in `src/inner_product_round.rs`, the notation hi_base_hi is used when the hi_base basis is split into two where the second hi

means higher part. This makes it even more confusing specifically when the h basis is also the g basis in other documents.

We recommend to use coherent notations in the code that correspond to the implementation's descriptions.

7.2 benchmark/

Commented parameters for full spectrum are not updated with new values, and has been done since then by Tari. The benchmarks were ran on our laptops and seemed coherent with the results presented in Tari's internal documentations [5].

7.3 transcript.rs

```
for item in &statement.minimum_value_promises {
    if let Some(minimum_value) = item {
        transcript.append_u64(b"vi - minimum_value", *minimum_value);
    } else {
        transcript.append_u64(b"vi - minimum_value", 0);
    }
}
```

The message of the label here is inconsistent with what is appended to the transcript: the minimum value is appended and not `vi - minimum value`.

We added a test to verify that the values were correctly unpacked by the transcript as the function calling the Merlin's transcripts function seemed like they were copying the same two values.

```
fn test_transcript_point_a_challenges_y_z() {
    ...
    let mut transcript = Transcript::new(b"test");
    let a: RistrettoPoint = RistrettoPoint::from_uniform_bytes(&buf);

    let (y, z) = crate::transcripts::transcript_point_a_challenges_y_z(&mut
    ↪  transcript, &a.compress()).unwrap();
    assert_ne!(y, z);
}
```

Overall, no issues were found on `transcript.rs` which is a good point.

7.4 transcript_protocol.rs

```
/// Compute a `label`ed challenge variable.
   fn challenge_scalar(&mut self, label: &'static [u8]) -> Result<Scalar,
   ↪  ProofError>;
```

```
   fn challenge_scalar(&mut self, label: &'static [u8]) -> Result<Scalar,
   ↪  ProofError> {
       let mut buf = [0u8; 64];
       self.challenge_bytes(label, &mut buf);
       let value = Scalar::from_bytes_mod_order_wide(&buf);
       if value == Scalar::ZERO {
           Err(ProofError::VerificationFailed(
               "Transcript challenge cannot be zero".to_string(),
           ))
       } else {
           Ok(value)
       }
   }
```

⚠ The comment on the signature of the function is confusing since no computation are involved there just the filling of a value.

👍 Overall, no issues were found on `transcript_protocol.rs` which is a good point.

7.5 bulletproof_gens.rs

```
/// # Extensible Generator Generation
///
/// Instead of constructing a single vector of size `m*n`, as described in the
↪  Bulletproofs paper, we construct each
/// party's generators separately.
///
/// To construct an arbitrary-length chain of generators, we apply SHAKE256 to a
↪  domain separator label, and feed each
/// 64 bytes of XOF output into the curve hash-to-group function. Each of the `m`
↪  parties' generators are
/// constructed using a different domain separation label, and proving and
↪  verification uses the first `n` elements of
/// the arbitrary-length chain.
///
/// This means that the aggregation size (number of parties) is orthogonal to the
↪  rangeproof size (number of bits),
/// and allows using the same `BulletproofGens` object for different proving
↪  parameters.
///
```

```
/// This construction is also forward-compatible with constraint system proofs,
↪   which use a much larger slice of the
/// generator chain, and even forward-compatible to multiparty aggregation of
↪   constraint system proofs, since the
/// generators are namespaced by their party index.
```

This technique is not detailed in Tari's internal specification. On the code, no bugs nor vulnerabilities were discovered during the time frame of the review.

We recommend Tari Labs to integrate this technique into their specifications.

Overall, no issues were found on `bulletproof_gens.rs` which is a good point.

7.6 pedersen_gen.rs

```
    /// Represents a pair of base points for Pedersen commitments
///
/// The Bulletproofs implementation and API is designed to support pluggable bases
↪   for Pedersen commitments, so that
/// the choice of bases is not hard-coded.
///
/// The default generators are:
///
/// * `h_base`: the curve basepoint;
/// * `g_base_vec`: the result of domain separated SHA3-512 (hash of unique
↪   indexed strings)
/// hash-to-group on input `B_bytes`.
#[derive(Clone, Debug, PartialEq)]
pub struct PedersenGens<P: Compressable> {
    /// Base for the committed value
    pub h_base: P,
    /// Compressed base for the committed value
    pub h_base_compressed: P::Compressed,
    /// Base for the blinding factor vector
    pub g_base_vec: Vec<P>,
    /// Compressed base for the blinding factor vector
    pub g_base_compressed_vec: Vec<P::Compressed>,
    /// Blinding factor extension degree
    pub extension_degree: ExtensionDegree,
}
```

A user of Tari's Bulletproof Plus implementation should always ensure that the g_base_vec comes from the hash of unique indexed strings. This should be stated in the specification and not in comment of the code.

Overall, no issues were found on `pedersen_gen.rs` which is a good point.

7.7 range_proof.rs

No bugs nor vulnerabilities were discovered and the code is compliant with the specification.

We noticed that in the provided example in comments of the code on the usage of the range proof functions, the blinding masks are all the same which we believe can be problematic and leaking unwanted information.

Overall, no issues were found on `range_proof.rs` which is a good point.

8 Resiliency tests

Resiliency tests performed dynamically in a Rust project play an important role in assessing and improving the project's resiliency against applicable threats. To that end, the auditors apply various failure scenarios and measure the code response, in order to evaluate the reliability, availability, and overall robustness, with a focus on the described threats (see Section 5.2). Such dynamic tests were also prioritize thanks to the warnings found during code quality assessment (see Chapter 4.4).

8.1 Fuzzing for Arithmetic Overflow

As depicted in Section 4.4, the `cargo clippy` command shown a numerous number of warnings, all of them regarding some potential overflows either in the underlying arithmetic or the array/slice indexes. While we checked a certain amount of them during code reviews, we still decided to make a fuzzer run during several days on the proper operations to see if we could cause any panics.

The applied fuzz-testing did not reveal any bugs nor overflows, which is a good point.

8.2 Playing with Nonce Generation

We used some internal tools to see if the nonce generation were behaving properly since they are a main point of interest in cryptographic protocols. Our test did not reveal, to the best of our knowledge, any flaws.

More specifically, we played a bit with the `nonce` function in src/utils/generic.rs that allows to generate a deterministic nonce from a label and optional two indexes.

The deterministic nonce generation seems to be properly done.

8.3 Fuzzing for difference in `curve25519-dalek`

In order to validate that the changes made to the `curve25519-dalek` crate do not impact the implementation of the underlying arithmetic for the Bulletproofs Plus protocols, we did a cartography of all the functions in the source code of Tari that comes from the `tari-curve25519` crate and does arithmetic operations. The list is the following:

- `multiscalar_`mul in src/generators/perdersen_gen.rs,
- `vartime_multiscalar_mul` in src/inner_product_round.rs,
- `batch_invert` in src/range_proof.rs.

For each of these functions, we fuzzed on both the Tari and original crate ones in order to catch a mismatch, namely we made differential fuzzing. After several days, we could not find any issue to the behavior of Tari crate with respect to original one.

We did not find any difference between the two Dalek crates in terms of input/output behavior.

9 Conclusion

To conclude, Quarkslab made a discovery of Tari's Bulletproofs Plus implementation for which a complete specification was produced. Based on the latter, an audit methodology and tests plans were defined in order to focus the security audit on relevant items within the allocated time frame. Using the defined security perimeter and tests, the audit unveiled some low and mostly informative issues in the codebase, but nothing critical or exploitable in the end. However, Quarkslab auditors note that for use case such as cryptographic protocols, every design choices should be reported in any kind of documentation in order for external users and reviewers to have a clear understanding of the underlying implementation.

Overall, it was a pleasure to work with Tari experts and maintainers on this audit, they were very helpful, security-minded, and willing to make the project more resilient.

Bibliography

[1] Tari Labs. *Tari Labs Bulletproofs Plus repository*. URL: https://github.com/tari-project/bulletproofs-plus (visited on Sept. 1, 2023) (cit. on p. 2).

[2] Heewon Chunga Kyoohyung Han Chanyang Ju Myungsun Kim and Jae Hong Seo. *BulletproofsPlus: Shorter Proofs for Privacy-Enhanced Distributed Ledger*. Cryptology ePrint Archive, Paper 2020/735. 2020 (cit. on pp. 2, 7, 16, 20, 23).

[3] Tari Labs. *Quarkslab Audit*. URL: https://github.com/tari-project/bulletproofs-plus/pull/91 (visited on Oct. 13, 2023) (cit. on p. 4).

[4] sowle and koe. *Zarcanum: A Proof-of-Stake Scheme for Confidential Transactions with Hidden Amounts*. Cryptology ePrint Archive, Paper 2021/1478. 2021 (cit. on pp. 6, 8, 16, 20).

[5] *RFC-0181/BulletproofsPlus*. URL: https://rfc.tari.com/RFC-0181$%5C_$BulletproofsPlus.html (visited on Aug. 3, 2023) (cit. on pp. 6, 8, 20, 23, 24, 26).

[6] Benedikt Bunz. *Bulletproofs - BPASE 18*. URL: https://www.youtube.com/watch?v=gZjDKgR4dw8 (visited on Sept. 1, 2023) (cit. on p. 7).

[7] Isis Agora Lovecruft and Henry de Valence. *Range proof for inner product*. URL: https://doc-internal.dalek.rs/bulletproofs/notes/range_proof/index.html (visited on Sept. 1, 2023) (cit. on p. 7).

[8] Cathie Yun. *Building on Bulletproofs*. URL: https://cathieyun.medium.com/building-on-bulletproofs-2faa58af0ba8 (visited on Sept. 1, 2023) (cit. on p. 7).

[9] Suyash Bagad. *Comparing Bulletproofs+ and Bulletproofs - Part I*. URL: https://suyash67.github.io/homepage/project/2020/07/03/bulletproofs_plus_part1.html (visited on Sept. 1, 2023) (cit. on p. 7).

[10] Henry de Valence. *Merlin*. URL: https://merlin.cool (visited on Sept. 1, 2023) (cit. on p. 10).

[11] Isis Agora Lovecruft and Henry de Valence. *curve25519-dalek*. URL: https://crates.io/crates/curve25519-dalek (visited on Sept. 1, 2023) (cit. on p. 10).

[12] Suyash Bagad, Omer Shlomovits, and Claudio Orlandi. *Monero BulletproofsPlus Security Audit*. URL: https://suyash67.github.io/homepage/assets/pdfs/bulletproofs_plus_audit_report_v1.1.pdf (visited on Sept. 1, 2023) (cit. on p. 16).

Appendix A

Appendix

A.1 Output of `cargo geiger`

```
qb@tari-bpp:~/bulletproofs-plus > cargo geiger --output-format Ascii
    Metric output format: x/y
    x = unsafe code used by the build
    y = total unsafe code found in the crate

Symbols:
    :) = No `unsafe` usage found, declares #![forbid(unsafe_code)]
    ?  = No `unsafe` usage found, missing #![forbid(unsafe_code)]
    !  = `unsafe` usage found

Functions  Expressions  Impls  Traits  Methods  Dependency

0/0        0/0          0/0    0/0     0/0      ?  tari_bulletproofs_plus 0.3.1
0/0        12/21        13/13  1/1     0/0      !  |-- blake2 0.10.6
0/0        0/0          0/0    0/0     0/0      :) |   `-- digest 0.10.7
0/0        16/16        0/0    0/0     0/0      !  |       |-- block-buffer 0.10.4
1/1        285/285      20/20  8/8     5/5      !  |       |   `-- generic-array
↪   0.14.7
0/0        5/5          0/0    0/0     0/0      !  |       |       |-- serde
↪   1.0.176
0/0        0/0          0/0    0/0     0/0      ?  |       |       |   `--
↪   serde_derive 1.0.176
0/0        15/15        0/0    0/0     3/3      !  |       |       |       |--
↪   proc-macro2 1.0.66
0/0        4/4          0/0    0/0     0/0      !  |       |       |       |   `--
↪   unicode-ident 1.0.11
0/0        0/0          0/0    0/0     0/0      ?  |       |       |       |--
↪   quote 1.0.32
0/0        15/15        0/0    0/0     3/3      !  |       |       |       |   `--
↪   proc-macro2 1.0.66
0/0        79/79        3/3    0/0     2/2      !  |       |       |       `-- syn
↪   2.0.27
0/0        15/15        0/0    0/0     3/3      !  |       |       |           |--
↪   proc-macro2 1.0.66
0/0        0/0          0/0    0/0     0/0      ?  |       |       |           |--
↪   quote 1.0.32
0/0        4/4          0/0    0/0     0/0      !  |       |       |           `--
↪   unicode-ident 1.0.11
0/0        0/0          0/0    0/0     0/0      :) |       |       |-- typenum
↪   1.16.0
1/1        24/24        0/0    0/0     0/0      !  |       |       `-- zeroize
↪   1.6.0
```

```
0/0        5/5         0/0    0/0     0/0      !  |       |          |-- serde
↪  1.0.176
0/0        0/0         0/0    0/0     0/0      :) |       |          `--
↪  zeroize_derive 1.4.2
0/0        15/15       0/0    0/0     3/3      !  |       |              |--
↪  proc-macro2 1.0.66
0/0        0/0         0/0    0/0     0/0      ?  |       |              |--
↪  quote 1.0.32
0/0        79/79       3/3    0/0     2/2      !  |       |              `-- syn
↪  2.0.27
0/0        0/0         0/0    0/0     0/0      :) |       |-- crypto-common 0.1.6
1/1        285/285     20/20  8/8     5/5      !  |       |   |-- generic-array
↪  0.14.7
0/0        2/2         0/0    0/0     0/0      !  |       |   |-- rand_core 0.6.4
3/7        71/228      1/1    0/0     3/3      !  |       |   |   |-- getrandom
↪  0.2.10
0/0        0/0         0/0    0/0     0/0      ?  |       |   |   |   |-- cfg-if
↪  1.0.0
1/60       10/502      0/2    0/0     5/50     !  |       |   |   |   `-- libc
↪  0.2.147
0/0        5/5         0/0    0/0     0/0      !  |       |   |   `-- serde
↪  1.0.176
0/0        0/0         0/0    0/0     0/0      :) |       |   `-- typenum 1.16.0
0/0        9/9         0/0    0/0     0/0      !  |       `-- subtle 2.5.0
0/1        176/193     0/0    0/0     0/0      !  |-- byteorder 1.4.3
0/0        0/0         0/0    0/0     0/0      ?  |-- derivative 2.2.0
0/0        15/15       0/0    0/0     3/3      !  |   |-- proc-macro2 1.0.66
0/0        0/0         0/0    0/0     0/0      ?  |   |-- quote 1.0.32
0/0        69/69       3/3    0/0     2/2      !  |   `-- syn 1.0.109
0/0        15/15       0/0    0/0     3/3      !  |       |-- proc-macro2 1.0.66
0/0        0/0         0/0    0/0     0/0      ?  |       |-- quote 1.0.32
0/0        4/4         0/0    0/0     0/0      !  |       `-- unicode-ident
↪  1.0.11
0/0        0/0         0/0    0/0     0/0      ?  |-- derive_more 0.99.17
0/0        0/0         0/0    0/0     0/0      ?  |   |-- convert_case 0.4.0
0/0        15/15       0/0    0/0     3/3      !  |   |-- proc-macro2 1.0.66
0/0        0/0         0/0    0/0     0/0      ?  |   |-- quote 1.0.32
0/0        69/69       3/3    0/0     2/2      !  |   `-- syn 1.0.109
0/0        0/0         0/0    0/0     0/0      :) |-- digest 0.10.7
0/0        0/72        0/3    0/1     0/3      ?  |-- itertools 0.6.5
0/0        14/14       0/0    0/0     0/0      !  |   `-- either 1.9.0
0/0        5/5         0/0    0/0     0/0      !  |       `-- serde 1.0.176
0/0        7/7         1/1    0/0     0/0      !  |-- lazy_static 1.4.0
0/0        5/5         0/0    0/0     0/0      !  |-- merlin 3.0.0
0/1        176/193     0/0    0/0     0/0      !  |   |-- byteorder 1.4.3
0/1        1/2         0/0    0/0     0/0      !  |   |-- keccak 0.1.4
0/0        2/2         0/0    0/0     0/0      !  |   |-- rand_core 0.6.4
1/1        24/24       0/0    0/0     0/0      !  |   `-- zeroize 1.6.0
0/0        32/32       0/0    0/0     0/0      !  |-- rand 0.8.5
1/60       10/502      0/2    0/0     5/50     !  |   |-- libc 0.2.147
0/2        0/20        0/1    0/0     0/0      ?  |   |-- log 0.4.19
0/0        5/5         0/0    0/0     0/0      !  |   |   `-- serde 1.0.176
```

```
0/0        0/0         0/0    0/0     0/0       ?  |   |-- rand_chacha 0.3.1
2/2        636/712     0/0    0/0     17/25     !  |   |   |-- ppv-lite86 0.2.17
0/0        2/2         0/0    0/0     0/0       !  |   |   |-- rand_core 0.6.4
0/0        5/5         0/0    0/0     0/0       !  |   |   `-- serde 1.0.176
0/0        2/2         0/0    0/0     0/0       !  |   |-- rand_core 0.6.4
0/0        5/5         0/0    0/0     0/0       !  |   `-- serde 1.0.176
0/0        2/2         0/0    0/0     0/0       !  |-- rand_core 0.6.4
0/0        5/5         0/0    0/0     0/0       !  |-- serde 1.0.176
0/0        0/0         0/0    0/0     0/0       :) |-- sha3 0.10.8
0/0        0/0         0/0    0/0     0/0       :) |   |-- digest 0.10.7
0/1        1/2         0/0    0/0     0/0       !  |   `-- keccak 0.1.4
0/2        151/802     0/0    0/0     0/0       !  |-- tari-curve25519-dalek 4.0.3
0/0        0/0         0/0    0/0     0/0       ?  |   |-- cfg-if 1.0.0
0/1        0/14        0/0    0/0     0/0       ?  |   |-- cpufeatures 0.2.9
0/0        0/0         0/0    0/0     0/0       ?  |   |-- curve25519-dalek-derive
↪  0.1.0
0/0        15/15       0/0    0/0     3/3       !  |   |   |-- proc-macro2 1.0.66
0/0        0/0         0/0    0/0     0/0       ?  |   |   |-- quote 1.0.32
0/0        79/79       3/3    0/0     2/2       !  |   |   `-- syn 2.0.27
0/0        0/0         0/0    0/0     0/0       :) |   |-- digest 0.10.7
0/0        2/2         0/0    0/0     0/0       !  |   |-- rand_core 0.6.4
0/0        5/5         0/0    0/0     0/0       !  |   |-- serde 1.0.176
0/0        9/9         0/0    0/0     0/0       !  |   |-- subtle 2.5.0
1/1        24/24       0/0    0/0     0/0       !  |   `-- zeroize 1.6.0
0/0        0/0         0/0    0/0     0/0       ?  |-- thiserror 1.0.44
0/0        0/0         0/0    0/0     0/0       ?  |   `-- thiserror-impl 1.0.44
0/0        15/15       0/0    0/0     3/3       !  |       |-- proc-macro2 1.0.66
0/0        0/0         0/0    0/0     0/0       ?  |       |-- quote 1.0.32
0/0        79/79       3/3    0/0     2/2       !  |       `-- syn 2.0.27
1/1        24/24       0/0    0/0     0/0       !  `-- zeroize 1.6.0

8/78       1623/3132   41/47  9/10    37/93

error: Found 4 warnings
```

A.2 Output of `cargo clippy`

```
    qb@tari-bpp:~/bulletproofs-plus > cargo clippy --no-deps -- -A clippy::all -W
    ↪  clippy::integer_arithmetic -W clippy::string_slice -W clippy::expect_used
    ↪  -W clippy::fallible_impl_from -W clippy::get_unwrap -W
    ↪  clippy::index_refutable_slice
     -W clippy::indexing_slicing -W clippy::match_on_vec_items -W
     ↪  clippy::match_wild_err_arm -W clippy::missing_panics_doc -W clippy::panic
     ↪  -W clippy::panic_in_result_fn -W clippy::unreachable -W
     ↪  clippy::unwrap_in_result -W clippy::unwrap_used
        Checking tari_bulletproofs_plus v0.3.1 (~/bulletproofs-plus)
    warning: lint `clippy::integer_arithmetic` has been renamed to
    ↪  `clippy::arithmetic_side_effects`
      |
      = note: requested on the command line with `-W clippy::integer_arithmetic`
```

```
warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
  --> src/generators/aggregated_gens_iter.rs:22:13
   |
22 |             self.party_idx += 1;
   |             ^^^^^^^^^^^^^^^^^^^
   |
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects
   = note: `-W clippy::arithmetic-side-effects` implied by `-W
   ↪  clippy::integer-arithmetic`

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
  --> src/generators/aggregated_gens_iter.rs:29:13
   |
29 |             self.gen_idx += 1;
   |             ^^^^^^^^^^^^^^^^^
   |
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
  --> src/generators/aggregated_gens_iter.rs:30:19
   |
30 |             Some(&self.array[self.party_idx][cur_gen])
   |                   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing
   = note: requested on the command line with `-W clippy::indexing-slicing`

warning: indexing may panic
  --> src/generators/aggregated_gens_iter.rs:30:19
   |
30 |             Some(&self.array[self.party_idx][cur_gen])
   |                   ^^^^^^^^^^^^^^^^^^^^^^^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
  --> src/generators/aggregated_gens_iter.rs:35:20
   |
35 |         let size = self.n * (self.m - self.party_idx) - self.gen_idx;
   |                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
   |
   = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
  --> src/generators/bulletproof_gens.rs:75:13
   |
75 |             g_vec[i].extend(&mut
↪  GeneratorsChain::<P>::new(&label).take(gens_capacity));
   |             ^^^^^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
  --> src/generators/bulletproof_gens.rs:78:13
   |
78 |             h_vec[i].extend(&mut
↪  GeneratorsChain::<P>::new(&label).take(gens_capacity));
   |             ^^^^^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
  --> src/inner_product_round.rs:86:90
   |
86 |         if !(hi_base.len() == n && ai.len() == n && bi.len() == n) ||
↪  (y_powers.len() != (n + 2)) {
   |
    ↪  ^^^^^^^
   |
   = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:111:36
    |
111 |             li: Vec::with_capacity(n * aggregation_factor + 2),
    |                                    ^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:112:36
    |
112 |             ri: Vec::with_capacity(n * aggregation_factor + 2),
    |                                    ^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:146:26
    |
146 |               let mut a1 = &self.gi_base[0] * r +
    |  _____________^
147 | |                 &self.hi_base[0] * s +
148 | |                 &self.h_base * (r * self.y_powers[1] * self.bi[0] + s
↪  * self.y_powers[1] * self.ai[0]);
    | |___________________________________________________^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/inner_product_round.rs:146:27
    |
146 |             let mut a1 = &self.gi_base[0] * r +
    |                           ^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:147:18
    |
147 |                 &self.hi_base[0] * s +
    |                  ^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:148:37
    |
148 |                 &self.h_base * (r * self.y_powers[1] * self.bi[0] + s *
↪  self.y_powers[1] * self.ai[0]);
    |                                     ^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:148:56
```

```
    |
148 |                 &self.h_base * (r * self.y_powers[1] * self.bi[0] + s *
↪  self.y_powers[1] * self.ai[0]);
    |                                                       ^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:148:73
    |
148 |                 &self.h_base * (r * self.y_powers[1] * self.bi[0] + s *
↪  self.y_powers[1] * self.ai[0]);
    |
     ↪  ^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:148:92
    |
148 |                 &self.h_base * (r * self.y_powers[1] * self.bi[0] + s *
↪  self.y_powers[1] * self.ai[0]);
    |
     ↪  ^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:149:25
    |
149 |             let mut b = &self.h_base * (r * self.y_powers[1] * s);
    |                         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/inner_product_round.rs:149:45
    |
149 |             let mut b = &self.h_base * (r * self.y_powers[1] * s);
    |                                             ^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing
```

```
warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:151:17
    |
151 |                 a1 += &self.g_base[k] * d[k];
    |                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/inner_product_round.rs:151:24
    |
151 |                 a1 += &self.g_base[k] * d[k];
    |                        ^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:151:41
    |
151 |                 a1 += &self.g_base[k] * d[k];
    |                                         ^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:152:17
    |
152 |                 b += &self.g_base[k] * eta[k]
    |                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/inner_product_round.rs:152:23
    |
152 |                 b += &self.g_base[k] * eta[k]
    |                       ^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
```

```
   --> src/inner_product_round.rs:152:40
    |
152 |                 b += &self.g_base[k] * eta[k]
    |                                        ^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:160:28
    |
160 |             self.r1 = Some(r + self.ai[0] * e);
    |                            ^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/inner_product_round.rs:160:32
    |
160 |             self.r1 = Some(r + self.ai[0] * e);
    |                                ^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:161:28
    |
161 |             self.s1 = Some(s + self.bi[0] * e);
    |                            ^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/inner_product_round.rs:161:32
    |
161 |             self.s1 = Some(s + self.bi[0] * e);
    |                                ^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:162:28
```

```
    |
162 |             let e_square = e * e;
    |                            ^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:164:30
    |
164 |                 self.d1.push(eta[k] + d[k] * e + self.alpha[k] *
↪  e_square)
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/inner_product_round.rs:164:30
    |
164 |                 self.d1.push(eta[k] + d[k] * e + self.alpha[k] *
↪  e_square)
    |                              ^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:164:39
    |
164 |                 self.d1.push(eta[k] + d[k] * e + self.alpha[k] *
↪  e_square)
    |                                       ^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:164:50
    |
164 |                 self.d1.push(eta[k] + d[k] * e + self.alpha[k] *
↪  e_square)
    |                                                  ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing
```

```
warning: slicing may panic
   --> src/inner_product_round.rs:171:19
    |
171 |         let a1 = &self.ai[..n];
    |                   ^^^^^^^^^^^^
    |
    = help: consider using `.get(..n)`or `.get_mut(..n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: slicing may panic
   --> src/inner_product_round.rs:172:19
    |
172 |         let a2 = &self.ai[n..];
    |                   ^^^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: slicing may panic
   --> src/inner_product_round.rs:173:19
    |
173 |         let b1 = &self.bi[..n];
    |                   ^^^^^^^^^^^^
    |
    = help: consider using `.get(..n)`or `.get_mut(..n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: slicing may panic
   --> src/inner_product_round.rs:174:19
    |
174 |         let b2 = &self.bi[n..];
    |                   ^^^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: slicing may panic
   --> src/inner_product_round.rs:175:27
    |
175 |         let gi_base_lo = &self.gi_base[..n];
    |                           ^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(..n)`or `.get_mut(..n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: slicing may panic
   --> src/inner_product_round.rs:176:27
```

```
    |
176 |         let gi_base_hi = &self.gi_base[n..];
    |                           ^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: slicing may panic
   --> src/inner_product_round.rs:177:27
    |
177 |         let hi_base_lo = &self.hi_base[..n];
    |                           ^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(..n)`or `.get_mut(..n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: slicing may panic
   --> src/inner_product_round.rs:178:27
    |
178 |         let hi_base_hi = &self.hi_base[n..];
    |                           ^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:179:30
    |
179 |         let y_n_inverse = if self.y_powers[n] == Scalar::ZERO {
    |                              ^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:184:13
    |
184 |             self.y_powers[n].invert()
    |             ^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:186:43
    |
186 |         let a1_offset = a1.iter().map(|s| s *
↪  y_n_inverse).collect::<Vec<Scalar>>();
    |                                           ^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:187:43
    |
187 |         let a2_offset = a2.iter().map(|s| s *
↪  self.y_powers[n]).collect::<Vec<Scalar>>();
    |                                           ^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/inner_product_round.rs:187:47
    |
187 |         let a2_offset = a2.iter().map(|s| s *
↪  self.y_powers[n]).collect::<Vec<Scalar>>();
    |                                               ^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:205:9
    |
205 |         self.round += 1;
    |         ^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:210:13
    |
210 |             c_l += a1[i] * self.y_powers[i + 1] * b2[i];
    |             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/inner_product_round.rs:210:20
    |
210 |             c_l += a1[i] * self.y_powers[i + 1] * b2[i];
    |                    ^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:210:28
    |
210 |             c_l += a1[i] * self.y_powers[i + 1] * b2[i];
    |                            ^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:210:51
    |
210 |             c_l += a1[i] * self.y_powers[i + 1] * b2[i];
    |                                                   ^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:211:13
    |
211 |             c_r += a2[i] * self.y_powers[n + i + 1] * b1[i];
    |             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/inner_product_round.rs:211:20
    |
211 |             c_r += a2[i] * self.y_powers[n + i + 1] * b1[i];
    |                    ^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:211:28
    |
211 |             c_r += a2[i] * self.y_powers[n + i + 1] * b1[i];
    |                            ^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:211:55
    |
211 |             c_r += a2[i] * self.y_powers[n + i + 1] * b1[i];
    |                                                       ^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:238:14
    |
238 |             &self.li[self.li.len() - 1].compress(),
    |              ^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:238:22
    |
238 |             &self.li[self.li.len() - 1].compress(),
    |                      ^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/inner_product_round.rs:239:14
    |
239 |             &self.ri[self.ri.len() - 1].compress(),
    |              ^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:239:22
    |
239 |             &self.ri[self.ri.len() - 1].compress(),
    |                      ^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:245:55
    |
245 |             P::mul_point_vec_with_scalar(gi_base_hi, &(e *
↪  y_n_inverse))?.as_slice(),
    |                                                       ^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:260:24
    |
260 |         let e_square = e * e;
    |                        ^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:261:32
    |
261 |         let e_inverse_square = e_inverse * e_inverse;
    |                                ^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/inner_product_round.rs:263:13
    |
263 |             self.alpha[k] += d_l[k] * e_square + d_r[k] *
↪  e_inverse_square;
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/inner_product_round.rs:263:30
    |
263 |             self.alpha[k] += d_l[k] * e_square + d_r[k] *
↪  e_inverse_square;
    |                              ^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:263:50
    |
263 |             self.alpha[k] += d_l[k] * e_square + d_r[k] *
↪  e_inverse_square;
    |                                                  ^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/inner_product_round.rs:263:13
    |
263 |             self.alpha[k] += d_l[k] * e_square + d_r[k] *
↪  e_inverse_square;
    |             ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: slicing may panic
  --> src/protocols/curve_point_protocol.rs:52:9
   |
52 |         (buffer[0..size]).copy_from_slice(&output.as_slice()[0..size]);
   |         ^^^^^^^^^^^^^^^^^
   |
   = help: consider using `.get(n..m)` or `.get_mut(n..m)` instead
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: slicing may panic
  --> src/protocols/curve_point_protocol.rs:52:44
   |
52 |         (buffer[0..size]).copy_from_slice(&output.as_slice()[0..size]);
   |                                            ^^^^^^^^^^^^^^^^^^^^^^^^^^
   |
   = help: consider using `.get(n..m)` or `.get_mut(n..m)` instead
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
  --> src/protocols/curve_point_protocol.rs:66:22
   |
66 |             out[i] = &point_vec[i] * *scalar;
```

```
   |                      ^^^^^^^^^^^^^^^^^^^^^^^
   |
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
  --> src/protocols/curve_point_protocol.rs:66:23
   |
66 |             out[i] = &point_vec[i] * *scalar;
   |                       ^^^^^^^^^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
  --> src/protocols/curve_point_protocol.rs:66:13
   |
66 |             out[i] = &point_vec[i] * *scalar;
   |             ^^^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
  --> src/protocols/curve_point_protocol.rs:79:22
   |
79 |             out[i] = &a[i] + &b[i];
   |                      ^^^^^^^^^^^^^
   |
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
  --> src/protocols/curve_point_protocol.rs:79:23
   |
79 |             out[i] = &a[i] + &b[i];
   |                       ^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
  --> src/protocols/curve_point_protocol.rs:79:31
   |
79 |             out[i] = &a[i] + &b[i];
   |                               ^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
  --> src/protocols/curve_point_protocol.rs:79:13
   |
79 |             out[i] = &a[i] + &b[i];
   |             ^^^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
  --> src/protocols/scalar_protocol.rs:54:22
   |
54 |             out[i] = scalar_vec[i] * scalar;
   |                      ^^^^^^^^^^^^^^^^^^^^^^
   |
   = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
  --> src/protocols/scalar_protocol.rs:54:22
   |
54 |             out[i] = scalar_vec[i] * scalar;
   |                      ^^^^^^^^^^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
  --> src/protocols/scalar_protocol.rs:54:13
   |
54 |             out[i] = scalar_vec[i] * scalar;
   |             ^^^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
  --> src/protocols/scalar_protocol.rs:65:22
   |
65 |             out[i] = a[i] + b[i];
   |                      ^^^^^^^^^^^
   |
   = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects
```

```
warning: indexing may panic
  --> src/protocols/scalar_protocol.rs:65:22
   |
65 |             out[i] = a[i] + b[i];
   |                      ^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
  --> src/protocols/scalar_protocol.rs:65:29
   |
65 |             out[i] = a[i] + b[i];
   |                             ^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
  --> src/protocols/scalar_protocol.rs:65:13
   |
65 |             out[i] = a[i] + b[i];
   |             ^^^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:234:43
    |
234 |         let mut a_li = Vec::with_capacity(bit_length *
↪  aggregation_factor);
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:235:43
    |
235 |         let mut a_ri = Vec::with_capacity(bit_length *
↪  aggregation_factor);
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:237:59
    |
237 |             let bit_vector = if let Some(minimum_value) =
↪  statement.minimum_value_promises[j] {
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:243:43
    |
243 |                     bit_vector_of_scalars(value - minimum_value,
↪  bit_length)?
    |                                           ^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:250:27
    |
250 |                 a_ri.push(bit_field - Scalar::ONE);
    |                           ^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:273:24
    |
273 |         let z_square = z * z;
    |                        ^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:276:47
    |
276 |         let mut y_powers = Vec::with_capacity(aggregation_factor *
↪  bit_length + 2);
```

```
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:278:21
    |
278 |         for _ in 1..(aggregation_factor * bit_length + 2) {
    |                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:279:27
    |
279 |             y_powers.push(y_powers[y_powers.len() - 1] * y);
    |                           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:279:27
    |
279 |             y_powers.push(y_powers[y_powers.len() - 1] * y);
    |                           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:283:40
    |
283 |         let mut d = Vec::with_capacity(bit_length * aggregation_factor);
    |                                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:287:20
    |
287 |             d.push(two * d[i - 1]);
    |                    ^^^^^^^^^^^^^^
    |
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:287:26
    |
287 |             d.push(two * d[i - 1]);
    |                          ^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:291:24
    |
291 |                 d.push(d[(j - 1) * bit_length + i] * z_square);
    |                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:291:24
    |
291 |                 d.push(d[(j - 1) * bit_length + i] * z_square);
    |                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:297:13
    |
297 |             *item -= z;
    |             ^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:300:13
    |
300 |             *item += d[i] * y_powers[bit_length * aggregation_factor -
↪  i] + z;
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:300:22
    |
300 |             *item += d[i] * y_powers[bit_length * aggregation_factor -
↪  i] + z;
    |                      ^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:300:29
    |
300 |             *item += d[i] * y_powers[bit_length * aggregation_factor -
↪  i] + z;
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:305:13
    |
305 |             z_even_powers *= z_square;
    |             ^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:307:17
    |
307 |                 *alpha1_val += z_even_powers * witness.openings[j].r[k]
↪  * y_powers[bit_length * aggregation_factor + 1];
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:307:48
    |
307 |                 *alpha1_val += z_even_powers * witness.openings[j].r[k]
↪  * y_powers[bit_length * aggregation_factor + 1];
    |                                                ^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:307:48
    |
307 |                 *alpha1_val += z_even_powers * witness.openings[j].r[k]
↪  * y_powers[bit_length * aggregation_factor + 1];
    |                                                ^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:307:75
    |
307 |                 *alpha1_val += z_even_powers * witness.openings[j].r[k]
↪  * y_powers[bit_length * aggregation_factor + 1];
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:359:37
    |
359 |         let (g_base_vec, h_base) = (statements[0].generators.g_bases(),
↪  statements[0].generators.h_base());
    |                                     ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:359:73
    |
359 |         let (g_base_vec, h_base) = (statements[0].generators.g_bases(),
↪  statements[0].generators.h_base());
    |
    ↪  ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪ https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:360:26
    |
360 |         let bit_length = statements[0].generators.bit_length();
    |                          ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪ https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪ side-effects
   --> src/range_proof.rs:361:26
    |
361 |         let mut max_mn = statements[0].commitments.len() *
↪ statements[0].generators.bit_length();
    |
    ↪ ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪ https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:361:26
    |
361 |         let mut max_mn = statements[0].commitments.len() *
↪ statements[0].generators.bit_length();
    |                          ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪ https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:361:60
    |
361 |         let mut max_mn = statements[0].commitments.len() *
↪ statements[0].generators.bit_length();
    |                                                                  ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪ https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:363:32
    |
363 |         let extension_degree =
↪ statements[0].generators.extension_degree();
```

```
    |                                ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:365:63
    |
365 |         if extension_degree !=
↪  ExtensionDegree::try_from_size(range_proofs[0].d1.len())? {
    |
    ↪  ^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:385:68
    |
385 |                 extension_degree !=
↪  ExtensionDegree::try_from_size(range_proofs[i].d1.len())?
    |
    ↪  ^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:389:16
    |
389 |             if statement.commitments.len() *
↪  statement.generators.bit_length() > max_mn {
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:390:26
    |
390 |                 max_mn = statement.commitments.len() *
↪  statement.generators.bit_length();
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:395:13
    |
395 |             statements[max_index].generators.gi_base_ref(),
    |             ^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:396:13
    |
396 |             statements[max_index].generators.hi_base_ref(),
    |             ^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:400:29
    |
400 |                 if value >> (bit_length - 1) > 1 {
    |                             ^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:411:21
    |
411 |                 if &statement_gi_base_ref[j] != gi_base_ref_item {
    |                     ^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:419:21
    |
419 |                 if &statement_hi_base_ref[j] != hi_base_ref_item {
    |                     ^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing
```

```
warning: indexing may panic
   --> src/range_proof.rs:478:37
    |
478 |         let (g_base_vec, h_base) = (statements[0].generators.g_bases(),
↪  statements[0].generators.h_base());
    |                                      ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:478:73
    |
478 |         let (g_base_vec, h_base) = (statements[0].generators.g_bases(),
↪  statements[0].generators.h_base());
    |
     ↪  ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:479:26
    |
479 |         let bit_length = statements[0].generators.bit_length();
    |                          ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:480:32
    |
480 |         let extension_degree =
↪  statements[0].generators.extension_degree() as usize;
    |                                ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:481:34
    |
481 |         let g_bases_compressed =
↪  statements[0].generators.g_bases_compressed();
    |                                  ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:482:33
    |
482 |         let h_base_compressed =
↪  statements[0].generators.h_base_compressed();
    |                                 ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:483:23
    |
483 |         let precomp = statements[max_index].generators.precomp();
    |                       ^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:489:13
    |
489 |             log_n += 1;
    |             ^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:496:31
    |
496 |             two_n_minus_one = two_n_minus_one * two_n_minus_one;
    |                               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:498:9
    |
498 |         two_n_minus_one -= Scalar::ONE;
    |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:510:35
    |
510 |         let mut msm_dynamic_len = extension_degree + 1;
    |                                   ^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:512:13
    |
512 |             msm_dynamic_len += item.generators.aggregation_factor() + 3
↪  + range_proofs[index].li.len() * 2;
    |
     ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:512:75
    |
512 |             msm_dynamic_len += item.generators.aggregation_factor() + 3
↪  + range_proofs[index].li.len() * 2;
    |
     ↪  ^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:546:33
    |
546 |             if 1 << li.len() != commitments.len() * bit_length {
    |                                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:552:30
    |
552 |             let gen_length = aggregation_factor * bit_length;
```

```
    |                               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:577:86
    |
577 |                     transcripts::transcript_points_l_r_challenge_e(&mut
↪  transcript, &proof.li()?[j], &proof.ri()?[j])?;
    |
    ↪  ^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:577:103
    |
577 |                     transcripts::transcript_points_l_r_challenge_e(&mut
↪  transcript, &proof.li()?[j], &proof.ri()?[j])?;
    |
    ↪  ^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:585:28
    |
585 |             let z_square = z * z;
    |                            ^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:586:28
    |
586 |             let e_square = e * e;
    |                            ^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:592:24
```

```
    |
592 |                 y_nm = y_nm * y_nm;
    |                        ^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:593:36
    |
593 |                 challenges_sq.push(challenges[i] * challenges[i]);
    |                                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:593:36
    |
593 |                 challenges_sq.push(challenges[i] * challenges[i]);
    |                                    ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:593:52
    |
593 |                 challenges_sq.push(challenges[i] * challenges[i]);
    |                                                    ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:594:40
    |
594 |                 challenges_sq_inv.push(challenges_inv[i] *
↪  challenges_inv[i]);
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:594:40
    |
594 |                 challenges_sq_inv.push(challenges_inv[i] *
↪  challenges_inv[i]);
    |                                       ^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:594:60
    |
594 |                 challenges_sq_inv.push(challenges_inv[i] *
↪  challenges_inv[i]);
    |
     ↪  ^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:596:26
    |
596 |             let y_nm_1 = y_nm * y;
    |                          ^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:599:25
    |
599 |             for _ in 0..bit_length * aggregation_factor {
    |                         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:600:17
    |
600 |                 y_sum += y_sum_temp;
    |                 ^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
     ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
```

```
   --> src/range_proof.rs:601:17
    |
601 |                 y_sum_temp *= y;
    |                 ^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:605:44
    |
605 |             let mut d = Vec::with_capacity(bit_length *
↪  aggregation_factor);
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:608:24
    |
608 |                 d.push(two * d[i - 1]);
    |                        ^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:608:30
    |
608 |                 d.push(two * d[i - 1]);
    |                              ^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:612:28
    |
612 |                     d.push(d[(j - 1) * bit_length + i] * z_square);
    |                            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:612:28
```

```
    |
612 |                     d.push(d[(j - 1) * bit_length + i] * z_square);
    |                            ^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:619:37
    |
619 |             let mut d_sum_temp_2m = 2 * aggregation_factor;
    |                                     ^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:621:25
    |
621 |                 d_sum = d_sum + d_sum * d_sum_temp_z;
    |                         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:622:32
    |
622 |                 d_sum_temp_z = d_sum_temp_z * d_sum_temp_z;
    |                                ^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:625:13
    |
625 |             d_sum *= two_n_minus_one;
    |             ^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:634:49
    |
634 |   ...                  let mut this_mask = (*d1_val -
    |  ____________________^
635 | | ...                      nonce(&seed_nonce, "eta", None, Some(k))? -
636 | | ...                      e * nonce(&seed_nonce, "d", None, Some(k))?)
↪  *
637 | | ...                      e_square.invert();
    | |_____________________^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:638:29
    |
638 | ...                  this_mask -= nonce(&seed_nonce, "alpha", None,
↪  Some(k))?;
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:640:33
    |
640 | ...                  this_mask -= challenges_sq[j] * nonce(&seed_nonce,
↪  "dL", Some(j), Some(k))?;
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:640:46
    |
640 | ...                  this_mask -= challenges_sq[j] * nonce(&seed_nonce,
↪  "dL", Some(j), Some(k))?;
    |                                   ^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:641:33
    |
641 | ...                  this_mask -= challenges_sq_inv[j] *
↪  nonce(&seed_nonce, "dR", Some(j), Some(k))?;
```

```
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:641:46
    |
641 | ...                   this_mask -= challenges_sq_inv[j] *
↪  nonce(&seed_nonce, "dR", Some(j), Some(k))?;
    |                                    ^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:643:29
    |
643 | ...                   this_mask *= (z_square * y_nm_1).invert();
    |                       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:665:29
    |
665 |                 let log_i = (32 - 1 - (i as u32).leading_zeros()) as
↪  usize;
    |                             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:667:24
    |
667 |                 s.push(s[i - j] * challenges_sq[rounds - log_i - 1]);
    |                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:667:24
    |
667 |                 s.push(s[i - j] * challenges_sq[rounds - log_i - 1]);
```

```
    |                         ^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:667:35
    |
667 |                 s.push(s[i - j] * challenges_sq[rounds - log_i - 1]);
    |                                   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:670:25
    |
670 |                 let g = r1 * e * y_inv_i * s[i];
    |                         ^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:670:44
    |
670 |                 let g = r1 * e * y_inv_i * s[i];
    |                                            ^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:671:25
    |
671 |                 let h = s1 * e * s[gen_length - i - 1];
    |                         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:671:34
    |
671 |                 let h = s1 * e * s[gen_length - i - 1];
    |                                  ^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:672:17
    |
672 |                 gi_base_scalars[i] += weight * (g + e_square * z);
    |                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:672:17
    |
672 |                 gi_base_scalars[i] += weight * (g + e_square * z);
    |                 ^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:673:17
    |
673 |                 hi_base_scalars[i] += weight * (h - e_square * (d[i] *
↪  y_nm_i + z));
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:673:65
    |
673 |                 hi_base_scalars[i] += weight * (h - e_square * (d[i] *
↪  y_nm_i + z));
    |                                                                  ^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:673:17
    |
673 |                 hi_base_scalars[i] += weight * (h - e_square * (d[i] *
↪  y_nm_i + z));
```

```
    |                 ^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:674:17
    |
674 |                 y_inv_i *= y_inverse;
    |                 ^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:675:17
    |
675 |                 y_nm_i *= y_inverse;
    |                 ^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:681:17
    |
681 |                 z_even_powers *= z_square;
    |                 ^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:682:32
    |
682 |                 let weighted = weight * (-e_square * z_even_powers *
↪  y_nm_1);
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:685:21
    |
685 |                     h_base_scalar -= weighted *
↪  Scalar::from(minimum_value);
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:690:13
    |
690 |             h_base_scalar += weight * (r1 * y * s1 + e_square * (y_nm_1
↪  * z * d_sum + (z_square - z) * y_sum));
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:692:17
    |
692 |                 g_base_scalars[k] += weight * d1[k];
    |                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: indexing may panic
   --> src/range_proof.rs:692:47
    |
692 |                 g_base_scalars[k] += weight * d1[k];
    |                                               ^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:692:17
    |
692 |                 g_base_scalars[k] += weight * d1[k];
    |                 ^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
```

```
   --> src/range_proof.rs:695:34
    |
695 |             dynamic_scalars.push(weight * (-e));
    |                                  ^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:697:34
    |
697 |             dynamic_scalars.push(-weight);
    |                                  ^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:699:34
    |
699 |             dynamic_scalars.push(weight * (-e_square));
    |                                  ^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:702:70
    |
702 |             dynamic_scalars.extend(challenges_sq.into_iter().map(|c|
↪  weight * -e_square * c));
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:704:74
    |
704 |             dynamic_scalars.extend(challenges_sq_inv.into_iter().map(|c|
↪  weight * -e_square * c));
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects
```

```
warning: indexing may panic
   --> src/range_proof.rs:713:34
    |
713 |             dynamic_scalars.push(g_base_scalars[k]);
    |                                  ^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: indexing may panic
   --> src/range_proof.rs:714:33
    |
714 |             dynamic_points.push(g_base_vec[k].clone());
    |                                 ^^^^^^^^^^^^^
    |
    = help: consider using `.get(n)` or `.get_mut(n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:815:42
    |
815 |         let mut buf = Vec::with_capacity(1 + (self.li.len() +
↪  self.ri.len() + 5 + self.d1.len()) * 32);
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:836:32
    |
836 |         if slice.is_empty() || (slice.len() - 1) % 32 != 0 {
    |                                ^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: slicing may panic
   --> src/range_proof.rs:841:71
    |
841 |         let extension_degree =
↪  ExtensionDegree::try_from(read_1_byte(&slice[0..])[0] as usize)?;
    |
    ↪  ^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:842:28
    |
842 |         let num_elements = (slice.len() - 1) / 32;
    |                            ^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:843:27
    |
843 |         if num_elements < 2 + 5 + extension_degree as usize {
    |                           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:848:43
    |
848 |         let num_inner_prod_vec_elements = num_elements - 5 -
↪  extension_degree as usize;
    |
    ↪  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: slicing may panic
   --> src/range_proof.rs:859:68
    |
859 |
↪  li.push(P::Compressed::from_fixed_bytes(read_32_bytes(&slice[1 + i *
↪  32..])));
    |
    ↪  ^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:859:74
    |
859 |
↪  li.push(P::Compressed::from_fixed_bytes(read_32_bytes(&slice[1 + i *
↪  32..])));
    |
    ↪  ^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:861:21
    |
861 |         for i in n..2 * n {
    |                     ^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: slicing may panic
   --> src/range_proof.rs:862:68
    |
862 |
↪  ri.push(P::Compressed::from_fixed_bytes(read_32_bytes(&slice[1 + i *
↪  32..])));
    |
    ↪  ^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:862:74
    |
862 |
↪  ri.push(P::Compressed::from_fixed_bytes(read_32_bytes(&slice[1 + i *
↪  32..])));
    |
    ↪  ^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:865:19
    |
865 |         let pos = 1 + 2 * n * 32;
    |                   ^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: slicing may panic
   --> src/range_proof.rs:866:64
    |
866 |         let a =
↪  P::Compressed::from_fixed_bytes(read_32_bytes(&slice[pos..]));
    |
    ↪  ^^^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: slicing may panic
   --> src/range_proof.rs:867:65
    |
867 |         let a1 =
↪  P::Compressed::from_fixed_bytes(read_32_bytes(&slice[pos + 32..]));
    |
    ↪  ^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:867:71
    |
867 |         let a1 =
↪  P::Compressed::from_fixed_bytes(read_32_bytes(&slice[pos + 32..]));
    |
    ↪  ^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: slicing may panic
   --> src/range_proof.rs:868:64
    |
868 |         let b = P::Compressed::from_fixed_bytes(read_32_bytes(&slice[pos
↪  + 64..]));
    |
    ↪  ^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing
```

```
warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:868:70
    |
868 |         let b = P::Compressed::from_fixed_bytes(read_32_bytes(&slice[pos
↪  + 64..]));
    |
    ↪  ^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: slicing may panic
   --> src/range_proof.rs:869:75
    |
869 |         let r1 =
↪  Option::from(Scalar::from_canonical_bytes(read_32_bytes(&slice[pos +
↪  96..])))
    |
    ↪  ^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:869:81
    |
869 |         let r1 =
↪  Option::from(Scalar::from_canonical_bytes(read_32_bytes(&slice[pos +
↪  96..])))
    |
    ↪  ^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: slicing may panic
   --> src/range_proof.rs:871:75
    |
871 |         let s1 =
↪  Option::from(Scalar::from_canonical_bytes(read_32_bytes(&slice[pos +
↪  128..])))
    |
    ↪  ^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing
```

```
warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:871:81
    |
871 |         let s1 =
↪  Option::from(Scalar::from_canonical_bytes(read_32_bytes(&slice[pos +
↪  128..])))
    |
    ↪  ^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: slicing may panic
   --> src/range_proof.rs:877:22
    |
877 |                     &slice[pos + 160 + i * 32..],
    |                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:877:28
    |
877 |                     &slice[pos + 160 + i * 32..],
    |                            ^^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
   --> src/range_proof.rs:900:32
    |
900 |         if slice.is_empty() || (slice.len() - 1) % 32 != 0 {
    |                                ^^^^^^^^^^^^^^^^^
    |
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: slicing may panic
   --> src/range_proof.rs:905:48
    |
905 |         ExtensionDegree::try_from(read_1_byte(&slice[0..])[0] as usize)
    |                                                ^^^^^^^^^^
    |
    = help: consider using `.get(n..)` or .get_mut(n..)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing
```

```
warning: indexing may panic
  --> src/range_witness.rs:27:32
   |
27 |         let extension_degree = openings[0].r_len()?;
   |                                ^^^^^^^^^^^
   |
   = help: consider using `.get(n)` or `.get_mut(n)` instead
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: arithmetic operation that can potentially result in unexpected
↪  side-effects
  --> src/utils/generic.rs:74:17
   |
74 |     if value >> (bit_length - 1) > 1 {
   |                 ^^^^^^^^^^^^^^^^
   |
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#arithmetic_side_effects

warning: slicing may panic
  --> src/utils/generic.rs:93:32
   |
93 |     buf32[..].copy_from_slice(&data[..32]);
   |                                ^^^^^^^^^^
   |
   = help: consider using `.get(..n)`or `.get_mut(..n)` instead
   = help: for further information visit
   ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: slicing may panic
   --> src/utils/generic.rs:100:31
    |
100 |     buf8[..].copy_from_slice(&data[..1]);
    |                               ^^^^^^^^^
    |
    = help: consider using `.get(..n)`or `.get_mut(..n)` instead
    = help: for further information visit
    ↪  https://rust-lang.github.io/rust-clippy/master/index.html#indexing_slicing

warning: `tari_bulletproofs_plus` (lib) generated 228 warnings
    Finished dev [unoptimized + debuginfo] target(s) in 0.55s
```