Security Audit Report

Axelar solana-chain-codec

Axelar smart contract

by ARDA

on April 15, 2026

Table of Contents

Disclaimer **3**

Terminology **4**

Objective **4**

Summary **5**

Code Issues & Recommendations **6**

C1: Unnecessary recovery byte conversion round-trip 6

C2: domain_separator in Multisig Prover instead of chain-codec as specified 8

Disclaimer

The report makes no statements or warranties, either expressed or implied, regarding the security of the code, the information herein or its usage. It also cannot be considered as a sufficient assessment regarding the utility, safety and bugfree status of the code, or any other statements.

This report does not constitute legal or investment advice. It is for informational purposes only and is provided on an "as-is" basis. You acknowledge that any use of this report and the information contained herein is at your own risk. The authors of this report shall not be liable to you or any third parties for any acts or omissions undertaken by you or any third parties based on the information contained herein.

Terminology

Code: The code with which users interact.

Inherent risk: A risk for users that comes from a behavior inherent to the code's design.

Inherent risks only represent the risks inherent to the code's design, which are a subset of all the possible risks. **No inherent risk doesn't mean no risk.** It only means that no risk inherent to the code's design has been identified. Other kind of risks could still be present. For example, the issues not fixed incur risks for the users, or the upgradability of the code might also incur risks for the users.

Issue: A behavior unexpected by the users or by the project, or a practice that increases the chances of unexpected behaviors to appear.

Critical issue: An issue intolerable for the users or the project, that must be addressed.

Major issue: An issue undesirable for the users or the project, that we strongly recommend to address.

Medium issue: An issue uncomfortable for the users or the project, that we recommend to address.

Minor issue: An issue imperceptible for the users or the project, that we advise to address for the overall project security.

Objective

Our objective is to share everything we have found that would help assessing and improving the safety of the code:

1. The **inherent risks** of the code, labelled R1, R2, etc.
2. The **issues** in the **code**, labelled C1, C2, etc.
3. The **issues** in the **testing** of the code, labelled T1, T2, etc.
4. The **issues** in the **other** parts related to the code, labelled O1, O2, etc.
5. The **recommendations** to address each issue.

Summary

Blockchain: Axelar

Initial scope

- **Repository:** https://github.com/axelarnetwork/axelar-amplifier
- **Commit:** `dde8792d56538acf5e51e433bd9c3e1c3454bf49`
- **Path(s):** `./contracts/chain-codec-solana`

Final scope

- **Repository:** https://github.com/axelarnetwork/axelar-amplifier
- **Commit:** `9ce604c2554536fcf46c8ea161a9bb2663a0f0bf`
- **Path(s):** `./contracts/chain-codec-solana`

0 inherent risk in the final scope

0 issue in the final scope

2 issues reported in the initial scope and 0 remaining in the final scope.

Severity	Reported			Remaining		
	Code	Test	Other	Code	Test	Other
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	0	0	0	0	0	0
Minor	2	0	0	0	0	0

Code Issues & Recommendations

C1: Unnecessary recovery byte conversion round-trip

Severity: Minor **Status:** Fixed

Location

`contracts/chain-codec-solana/src/solana.rs`

Description

The ECDSA recovery byte goes through an unnecessary round-trip conversion across the Axelar stack for Solana. The full cycle is as follows:

1. Ampd builds ECDSA signatures with the standard recovery byte in range {0, 1}. This is the standard secp256k1 recovery ID range.

2. The Solana chain-codec converts the recovery byte to the Ethereum range {27, 28} (link):

```
// In chain-codec-solana/src/solana.rs
signature[64] = signature[64]
    .checked_add(27)
    .ok_or(Error::InvalidSignature)?;
```

3. The Solana Gateway converts it back to range {0, 1} for use with `secp256k1_recover` (link):

```
// In solana-axelar-gateway/src/state/verification_session.rs
let recovery_id = if *recovery_id == 27 || *recovery_id == 28 {
    recovery_id.saturating_sub(27)
} else {
    return false;
};
```

This round-trip conversion ({0,1} → {27,28} → {0,1}) is unnecessary, because the Ethereum-style recovery IDs {27, 28} serve no purpose on Solana. The Sui integration, for comparison, works directly with the standard {0, 1} range without any intermediate conversion.

Recommendation

We recommend removing the `checked_add(27)` conversion in the Solana chain-codec and the corresponding `saturating_sub(27)` back-conversion in the Solana Gateway, so that the recovery byte stays in the standard {0, 1} range all along.

C2: domain_separator in Multisig Prover instead of chain-codec as specified

Severity: Minor **Status:** Fixed

Description

The ARC-11 specification states that the `domain_separator` should be moved from the Multisig Prover into the chain-codec contract:

"The domain_separator (as shown in the EncoderExt trait above) will be moved into the chain-codec contract, as it is not used anywhere else in the multisig-prover contract."

However, the `domain_separator` is still present and used in the Multisig Prover config (link).

Recommendation

We recommend moving the `domain_separator` from the Multisig Prover into the chain-codec contract as specified in ARC-11.