Axelar Security Overview

This page is adapted from a post originally published to axelar.network/blog.

Security is a multidimensional problem with binary outcomes. Defining the security of a system is an incredibly complex task, but with rigorous models, designs and engineering practices, it's possible to build systems that can stand against attacks. When it comes to securing cross-chain communication, there are two core properties.

- Safety: how many nodes in the system does an adversary need to compromise to double-spend or maliciously execute a transaction (that may lead to an invalid asset transfer, for instance)?
- Liveness: how many nodes in the system does the adversary need to compromise to halt the system?

The goal is to achieve the above properties with maximal robustness of the network. This is achieved by Axelar, by implementing multiple different layers of security features.

Proof-of-stake consensus

In Byzantine systems, if you break liveness, the system halts its operation, but an adversary would need to do much more to break safety. For instance, a blockchain may halt under a denial-of-service attack on its validators, limiting their ability to send messages and reach consensus. But to break the safety of the blockchain, the adversary might need to find vulnerabilities in validator nodes and extract their secret keys. Robustness of such systems is thus achieved via diversification of software deployments. One validator might deploy on an Ubuntu machine, another on Debian, some in cloud environments, and some on-premise.

This kind of decentralization is critical to satisfy strong safety and liveness properties needed to handle cross-chain functions. This is why Axelar is built using battle-tested proof-of-stake consensus, supporting a diverse and dynamic validator set. Anyone can join, anyone can participate, anyone can contribute to the security of the network.

Quadratic voting

Concentration of power is a problem that many proof-of-stake networks today face. Validators that do a good job operating their nodes tend to receive more delegations from the community and have higher stake in the system. Standard proof-of-stake systems use delegated weight to count how many votes a validator should have in the system. The unfortunate side effect of this is concentration of voting power: if a single validator has a lot of stake, they have a lot of influence in the system.

Even with a system that is technologically capable of decentralization, it's necessary to determine a mechanism to increase decentralization over time, and counteract concentration of power. By implementing quadratic voting, Axelar is breaking new ground in proof-of-stake security. Quadratic voting has been widely discussed among blockchain leaders, but to date no major blockchain has implemented it.

Quadratic voting is a way to continue decentralizing the system while making it harder for validators to accumulate voting power. The number of votes each validator may cast is equivalent to the square root of their stake. A threshold of validators, weighted by the square root of the stakes, then must collectively coauthorize every cross-chain request. As validators accumulate stake, it becomes more difficult for them to accumulate voting power.

Note that while quadratic voting is implemented for validation and processing of cross-chain transactions, the underlying consensus follows the standard PoS staking model (1 token, 1 vote) for block production and rewards.

Preemptive measures

No system, no matter how well designed, is invulnerable. In order to best react in case of a failures, whether malicious or accidental, Axelar network implements a number of additional safety measures.

Rate limits

Axelar network supports General Message Passing, but certain major assets (e.g., USDC, FRAX, ETH) are implemented as "routing" assets on the network, which allows applications to send tokens with messages in one function. These assets are subsequently locked and minted across the Axelar gateways. The gateways have a rate limiting function, putting a cap on how much of each asset can be transferred in a given time interval.

Key rotations on the network

Validators of the Axelar network maintain keys that allow them to coauthorize cross-chain requests, a process similar to how validators coauthorize every block on the blockchain. Validators are strongly encouraged to implement best practices for isolating these keys, and arerequired to rotate them periodically . These key rotations secure the network against a persistent attacker, who might try to accumulate malicious keys by compromising validators sequentially.

Audits

The Axelar network code periodically goes through rigorous audits. Audits are published here .

In addition, all Axelar network code and contracts are open-source, meaning anyone can review the code and look for vulnerabilities. Comments and revision from white-hat hackers are actively encouraged. There is also an active bug-bounty program set up, which incentivizes security submissions.

Application-level security add-ons

On top of the above security measures, Axelar supports Turing-complete General Message Passing . This functionality enables application developers to build their own custom cross-chain security policies. For instance, a DeFi application may impose additional limits on the volume of transferred funds, repeat transactions, co-authoring large transfers, etc. These application-specific features can work alongside the network requirements to deliver an even stronger security guarantee.

Summary conclusion

In summary, Axelar tackles the complex issue of network security by having multiple layers of defenses in place. The security stack begins with decentralized, proof-of-stake consensus and diverse node tech stacks. Quadratic voting supports continued, scalable decentralization over time by preventing power consolidation among the top validators. To contain any potential errors and other failures that may occur, preventative measures are put in place, like frequent key rotations and transaction rate limits. The codebase has been thoroughly audited and is open source, with a bug bounty program to encourage whitehat submissions from the community. General Message Passing provides the ability for application-layer projects to add security features as needed for their use case, atop the Axelar security stack described here.

Edit this page

On this page * Proof-of-stake consensus * Quadratic voting * Preemptive measures * * Rate limits * * Key rotations on the network * * Audits * Application-level security add-ons * Summary conclusion