Team name

TNLS

Project Description

In this proposal, we propose a bridge solution for encrypted data between SCRT and public blockchains. We review existing bridge solutions and their respective tradeoffs, and we outline the need for an encrypted data bridge. We then describe the details of our solution, which uses similar routing and security techniques that public to public data bridges use but adds in industry-grade encryption techniques to ensure that data remain private throughout the entire process—with routers and validators ensuring the integrity of the private data via consensus. We close with a discussion of potential challenges and pitfalls of our solution and our mitigations.

Problem / Solution

The multi-chain future is no longer an abstract vision. It has become reality—and a painful one at that, since cross-chain solutions remain clunky from a user perspective and perilous from a security perspective. Although many people have expressed concerns about the inherent security limitations of bridges, it is clear that, at least in the short to medium term, bridges serve a clear infrastructure need in the market.

Current bridges focus on providing users with liquidity to easily move assets across chains and layer-2s, and more and more liquidity bridges are beginning to generalize their solutions to provide support for data and message passing to enable true multi-chain apps to be built. The bridging landscape is crowded, indeed, and many of these protocols are beginning to converge on similar architectures and solutions—Wormhole, Synapse, Layer Zero, Axelar, Celer, Connext, and more. Notwithstanding all of the potential bridging options for both liquidity and data, there is still one need that existing solutions do not serve: bridging data privately.

More specifically, there is a need for an encrypted bridge that passes data between public blockchains (like Ethereum) and private blockchains that are optimized for general-purpose computation (like Secret Network and Oasis). There are a variety of use cases that private computation-focused blockchains serve, but until now they have been limited by their siloed nature.

More specifically, applications must be built either fully atop these private blockchains—which often presents efficiency and scaling challenges—or, alternatively, applications can be built on public blockchains but can use private blockchains only for specific computational purposes. The latter is clearly superior for efficiency, security, and composability reasons. But until now the former has been the dominant mode of development for private blockchains because there is no true private data bridge from public chains to private chains—a bridge that would allow only private computation to occur on the private chain without revealing any of the data in the bridging process.

Sample use cases include:

Interchain KYC that preserves privacy, similarly with credit score checking

On-chain non-frontrunnable investment portfolios that run on the larger public chains

NFTable data, such that the data itself is accessible only via privacy-preserving contracts but the NFTs of that data can be sold via public chains/with all the existing NFT infra. (Legendao interfaces with Eth/Sol for free, etc)

This is where our solution comes in. Our bridge is optimized for communication between private and public chains (and, if necessary, between private and private chains). Our bridge uses similar routing and security techniques that public to public data bridges use but adds in industry-grade encryption techniques to ensure that data remain private throughout the entire process—with routers and validators ensuring the integrity of the private data via consensus.

More specifically, our solution works as follows. It is based on variants of the Axelar and Chainlink architectures for public-public contract calling, with a master gateway on each connected chain and a relayer network called TNLS connecting the gateways. For our cryptographic systems, we'll be using industry standards similar to those found in TLS/SSL, i.e. elliptic curve based cryptography for key exchange, AES for actual encryption, serving as a hybrid system (basically, PKCS #7

). Additionally, in order to ensure message veracity, every time the system transfers between gateways, from the initial user, or from contract to gateway, it's signed by the gateway it just left using ECDSA variants.

A short version of our protocol is as follows: "to call a private contract from a public one, you call the master public gateway with user-signed and encrypted inputs, it passes the inputs to TNLS, TNLS passes them to the master private gateway, which checks the signature then passes them on to the desired private contract, that contract decrypts encrypted inputs, executes user code, signs the result, and outputs the signed result back to the master private gateway, then the master private gateway verifies the signature, signs the result itself, and passes it back to TNLS, which passes it back to the master public gateway, which verifies the private gateway signature then calls the callback of the original contract".

A schema image is below:

[

Untitled-2022-04-05-1156

1440×350 48.6 KB

](https://global.discourse-cdn.com/standard17/uploads/enigma1/original/2X/7/781961dced1934b4e78ba1779adb4cde061ca658.png)

TNLS, our relayer network, is relatively architecturally simple — it's a DPOS based architecture where nodes get rewarded for the total number of passed messages/transactions per block based on the amount they have staked, and all nodes are doing initially is passing messages from one chain to another (initially via API, though this can be revisited). The network will become more complex as we add the capacity to programmatically add new gateways, where we'll need to have some governance capabilities to support that gateway addon so fake gateways don't get added. The gateways themselves serve both to package up calls between chains and to verify that fake tasks aren't being passed off as real ones/tasks aren't being misdirected.

The primary additional costs of this solution are the additional signatures and verifications inherent in packet-passing. Efficiency-wise, we get the upside of trusting private signing keys on both ends of the communications chain (one from the user and one from the private compute source) so we don't need to worry about multisig on the relayer network.

Go-to-Market plan

Our GTM is to work with devs who've already indicated interest in building atop TNLS – this is infra that's needed for several use cases, so we'll launch with those use cases ideally (secret Bazaar style)

Team members and experience

- Leor Fishman – Leor has worked in software for ~7 years, and managed teams for 3 – he's led large-scale intersystem communications projects before and worked on E2E encryption-based protocols pretty similar to this. He's also been close with the SCRT team since ~2017 and been contributing to specs/security issues for those 5 years.
- Ben Simon – Ben has been in the crypto space for ~3 years, and ran research at Mechanism for 2 of those years before moving to Spark where he's now a venture partner. He's also run several defi protocols before.

github

- FishmanL (Leor Barak Fishman) · GitHub

LinkedIn

- https://www.linkedin.com/in/leor-f-63490785/
- https://www.linkedin.com/in/benjamin-simon-56826b120/

Roadmap and Ask

Roadmap: Our path to testnet has 3 substeps:

one at complete approved spec and architecture design (with proof of security) (3 months),

one at a version of this working on testnet with a dev machine serving as the sole node (3 months),

final with this working on testnet with a testnet of validator nodes running (2 months).

Post testnet, our path more generally has the following steps:

1. get to testnet: ~8 months
2. get to mainnet for a pair of chains (SCRT-ETH or SCRT-SOL): ~4 months
3. Improve the chain adding/routing infrastructure/add more gateways: ~2 months per dev per chain, potentially less

4. Add specialized gateways for unique computation (Zama, etc): 3-6 months for a pair of devs, probably? (since it'll be more important to add caching for routing, etc).

We will require 12 months to complete this project to mainnet. We anticipate needing 3 full time developers to do so at 200k annual salary (100 an hour), for a total grant amount of 600k USD or 115k SCRT.

Planned milestones:

- Months 1-3: Completed architectural spec + basic gateways complete
- Months 3-6: Testnet with dev machine serving as node, passing arbitrary contracts. Audits on gateways performed here.
- Months 6-9: Testnet with validator nodes, passing arbitrary contracts. Relay network audits/attacks performed here.
- Months 9-12: Launched network!

Ideally, we can receive payments in 3 disbursements, one at the start of the work, one at completion of dev machine-based testnet, one at network launch.

We would be willing to consider part payment in SCRT, up to 80 percent.