Sygma

Sygma Litepaper

Build Secure Cross-Chain Applications

Published July 2022

Introduction

In recent years, dozens of new blockchain ecosystems have emerged and gained traction. However, dApp development remains siloed. While applications are deploying to different chains, the same app on different networks is a separate entity.

This means that, for most users, doing things across different ecosystems is complicated. Sygma addresses this by allowing dApp developers to bring cross-network functionality and a seamless UX to their users. All without needing to run a trusted relayer set, removing security from a dApp developer's concern. Security is something Sygma provides natively.

Sygma's Origin Story

Three years ago, there were few trustworthy cross-chain bridging protocols. Since then, the open-source ChainBridge cross-chain repo emerged as one of the first comprehensive multi-directional bridges for EVM and Substrate-based chains. ChainBridge has have been adopted and relied upon by a number of leading blockchains, transferring over $600 million in bridged assets since inception.

After years of working on bridging solutions, we're battle-tested and have learned a lot of valuable lessons along the way. In this sense, Sygma is the product of all our experiences. Like the Greek letter sigma (Σ), Sygma alludes to summation, the adding up of many things.

We chose this name because Sygma will allow developers to interweave blockchain protocols in a way that gives users something greater than the sum of the parts. Sygma takes the notion of bridging one step further—from simple point-to-point money transfers to an extensible, any-to-any communication protocol.

Building with Sygma: Use Cases

Sygma is easy to configure and compatible with many blockchain networks. It's designed to transfer compatible token standards such as ERC20 tokens and ERC721 tokens (NFTs).

More importantly, Sygma supports **generic message passing**, enabling builders on one chain to call any function on any other connected chain. This dramatically increases the number of options at a developer's disposal.

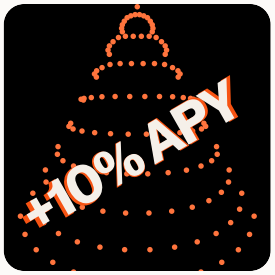
Building with Sygma: Use Cases

For instance, with Sygma, users can:

Borrow on one chain with collateral from another chain

Market NFTs seamlessly on any chain

Bridge directly from their wallet or portfolio manager



Use a yield aggregator to farm on any chain without moving funds

Vote on DAO governance proposals from any chain

Sygma protocol design

Sygma design was influenced by EVM standards and generally allows the transfer of any type of message. A Sygma message is a smart contract function call packed using [4-byte method signature] encoding. The message includes a list of desired parameters. And the Sygma relayer network is responsible for delivering those messages.

Because of the current nature of the smart-contract interactions, the market ultimately forced dApp developers to adopt standards, resulting in certain types of messages being predefined purely for convenience. That's why Sygma, from the start, supports ERC20 and ERC721 contract interfaces, allowing dApp builders who rely on these standards to fast-track their development cycle. Beyond those predefined interfaces, Sygma allows you to easily implement any type of application-specific messaging—the possibilities are infinite.

Sygma currently works on EVM-based networks but can easily be extended to Substrate, Cosmos, etc. All the interactions with Sygma happen on-chain making it easy to start bridging without deploying additional infrastructure. Usage of the Sygma SDK and a mindful approach to designing application-specific contracts are all that is required to start building.

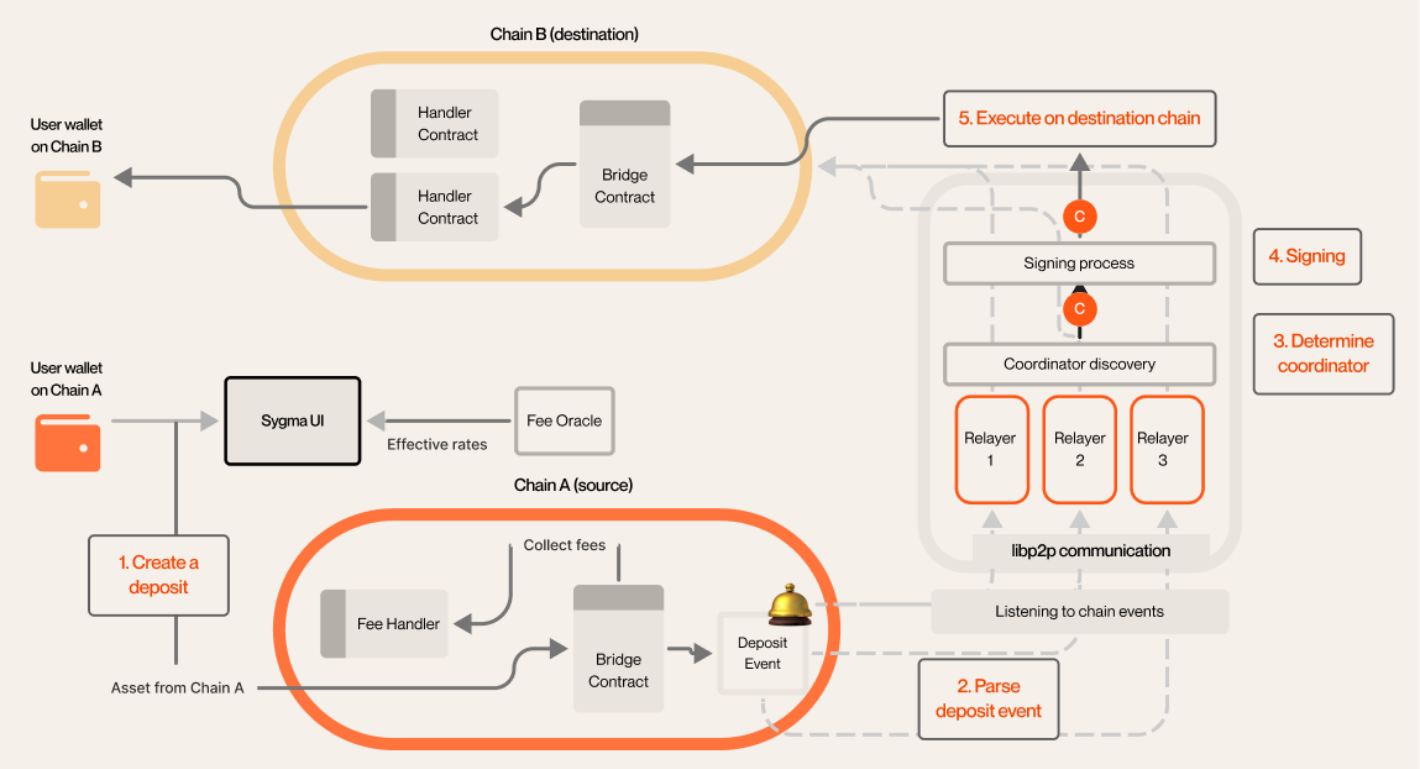
This allows for a two-pronged benefit of convenience as well as future-proofing. For instance, application-specific chains and dApps can focus on their core competencies, and when new chains are available, developers can easily tap into these network effects.

Sygma employs a multi-party computation (MPC) model, including a number of trusted relayer nodes. Under the secure MPC model, these trusted relayer nodes will be run by reliable entities in the web3 space. Distribution among these independent entities spreads responsibility and mitigates the risk that any single relayer acts unfairly or maliciously.

Components

Relayer Set

The “relayer” is an off-chain component designed to listen to deposit events on connected blockchains. When an event is detected on the source network, the relayer sends the corresponding transaction to be executed on the destination network. Here’s how it looks:



To protect against a corrupted relayer, execution is permitted only when consensus is achieved among the majority of relayers. This process involves an off-chain signature aggregation and voting based on the Threshold Signature Cryptographic Algorithm. Which means:

- **Lower fees** on the destination network, as only one transaction is required to process the transfer.
- The private key is not stored on any of the relayers. Relayers store only part of the private key used to sign the transaction. The full private key never appears on any relayer or at any step of the voting. To gain access to the private key, the majority of relayers would have to collude or be otherwise compromised.

Under the initial implementation, the set of relayers is a configured list run by our partners. In this way, the system is protected against Sybil attacks. As an additional benefit, this removes the need to build complex governance solutions to manage the system out of the gate.

Smart Contracts

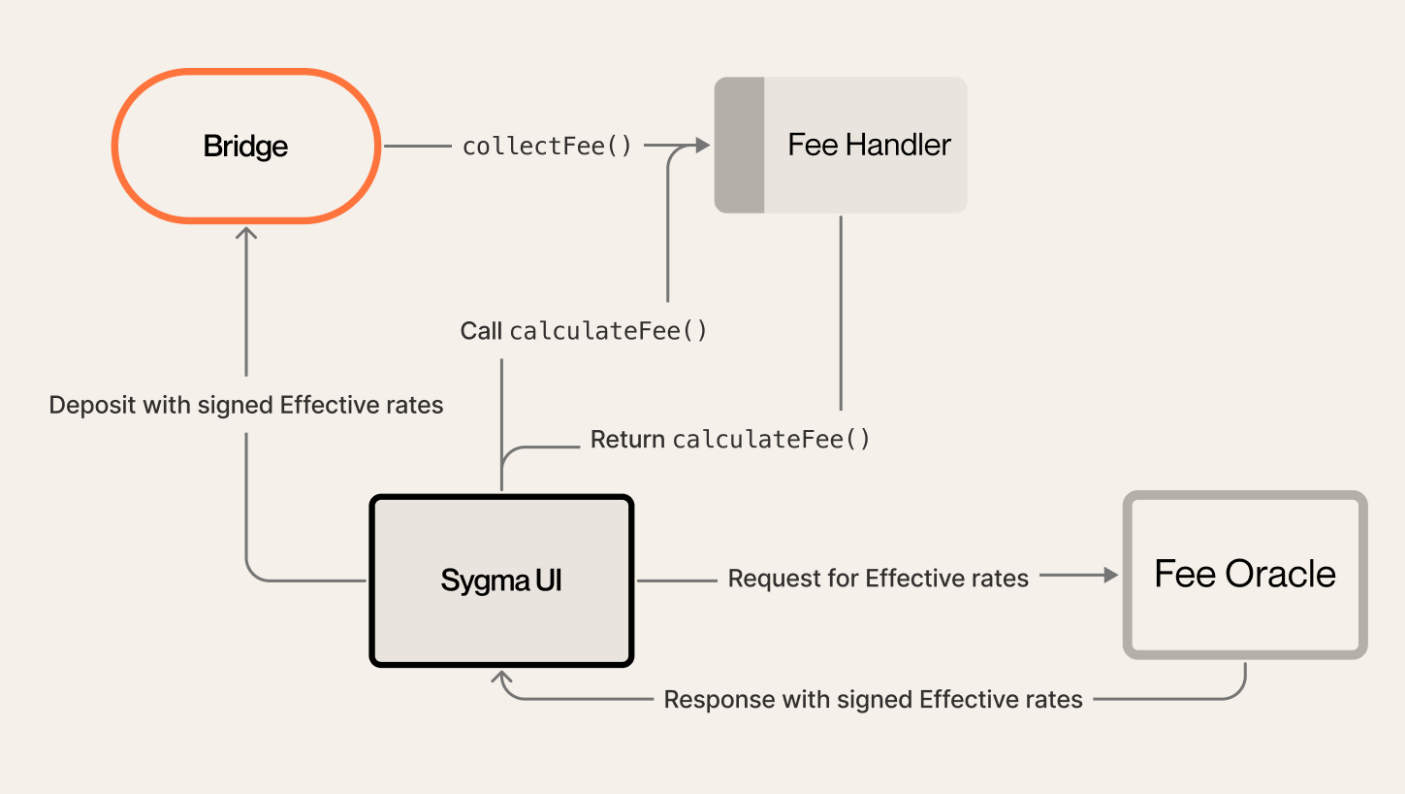
Smart contracts are the on-chain aspect of Sygma, which are composed of:

- The **Core** serves as the skeleton of our protocol and is the system's core component in that it accepts transactions on the destination networks. At the same time, detailed implementation is stored in the Protocol Handler.
- The **Protocol Handler** is the exact implementation of the logic that allows specific data to be executed on the destination network. For example, for general ERC20 and ERC721 token transfers, we are using ERC20Handler and ERC721Handler accordingly. For Generic data transfers, we use genericHandler. One ERC20Handler can be used for all the token transfers if they meet the ERC20 standard, and the same goes for existing ERC721 or any other future specific Protocol Handlers.
- The **Fee Handler** manages the business logic of fees inside Sygma. This logic decides the fee percentage and which currency (base or token) is being transferred based on transfer parameters (e.g., source and destination networks, token type, etc.). The only external data that Fee Handlers need are the effective rates.

Fee Oracle

The Fee Oracle is a service that provides signed effective rate information to the Fee Handler. This rate is used to calculate the transaction fees on the destination network that should be covered by the service fee. Effective rates consist of the gas price of the destination network and the exchange rates of the currency being transferred to the base currency of the destination network.

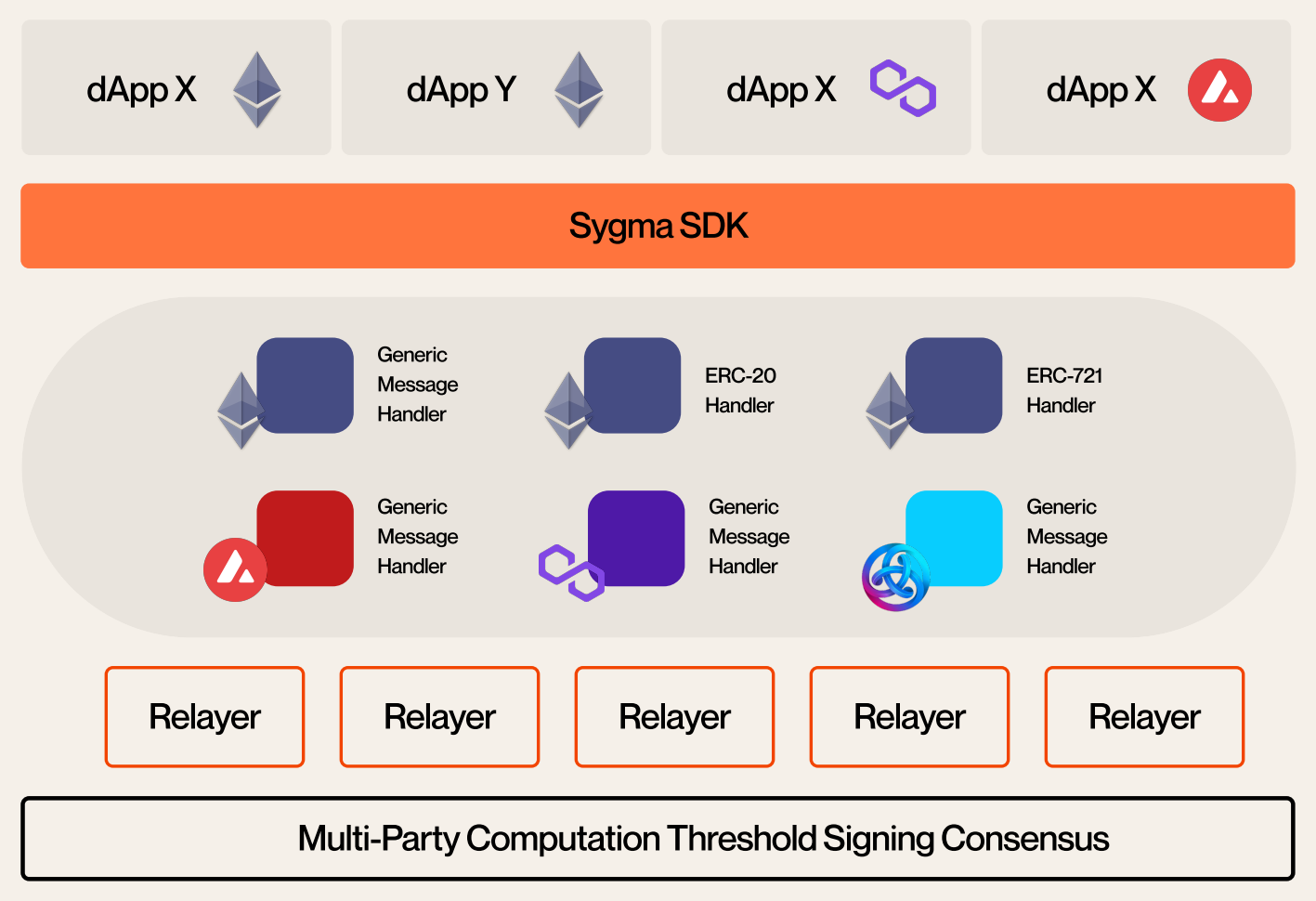
This information is used to determine the cost of transactions on the destination network and adjust the service fees. When the gas price of the destination network changes dramatically, it allows dynamic fee adaptation with a more fair service fee for users.



At launch, the Fee Oracle service will be centralized. The current architecture implies a future update of the Fee Oracle to a more decentralized and trustless solution. However, since the calculated fee is visible and approved by the user, the current Fee Oracle does not pose a security threat to the entire system.

Software development kit

Sygma makes it easy for developers to add interoperability to their dApp or protocol with minimal overhead. Rather than taking the time and energy to host your own bridge, Sygma will develop and maintain the infrastructure while providing application-specific JavaScript SDKs, support, and documentation. These SDKs are designed to make #buildingwithsygma easy.



User Interface

To enable communication with Sygma, we've created a set of core UI components. There's the **Primary UI**, a simple interface for sending transactions. The main UI uses MetaMask or Wallet Connect to connect a user's account, allowing users to send ERC20 tokens, estimate service fees, follow the transfer execution timeline, etc.

There's also **Explorer**, an interface that makes it possible to observe and track all the transfers within the Sygma ecosystem. Plus, we've created a JavaScript **Widget UI** that enables developers to easily inject the bridging UI into any website.

Launch

Pilot Program

The Sygma pilot program aims to work directly with teams on cross-chain use cases and provide support as you implement your vision.
If you're interested in joining, please fill out this short form.

Closed Beta

Initially, Sygma will be deployed on selected networks and will be available to dApps participating in our pilot program.

Public Launch

After initial feedback and fine-tuning, we will open Sygma and its accompanying SDK to all dApp developers. The relayer set will be expanded to include new partners.

Learn more about Sygma

Website | Repo | Discord