TL;DR

We propose Apsis, a blockchain that supports synchronous and reliable invocation to HTTPS APIs with retroactive verifiability. Apsis can be used to build on-chain Oracles which provides end-to-end data verifiability from data sources to data consumers. It can be also used to build hybrid blockchain applications to take advantage of both Web 2.0 and Web 3.0.

Background

Current blockchain applications rely on so-called off-chain Oracles to access external data, such as asset price in centralized exchanges. Existing Oracle solutions, such as ChainLink, API3 and Band are all off-chain Oracles since they have off-chain segments on their data path between data providers and data consumer DApps.

Off-chains Oracles are the security bottleneck of blockchain applications. They share similar drawbacks:

1. No off-chain verifiability: Data consumers have no way to validate whether the data consumed is generated from the target data source;
2. Over reliance on token economy: Oracles have to overpay node operators to incentivize honest operations;
3. Pull vs Push: Pull-based Oracle offers more up-to-date data but forces asynchronous interactions. Push-based Oracle only works for known dataset or known APIs, and its data is often stale.

We believe that off-chain Oracle is a compromise and temporary solution to access external data. If a blockchain itself can access HTTP API persistently and reliably, blockchain applications can read external data in transactions instead of trusting an external off-chain party to feed them the correct data. Moreover, the application scope of blockchain can be significantly expanded with HTTPS access capabilities.

Apsis Consensus

Blockchain data is retroactively verifiable, but HTTP data is not. However, this could be achieved if the follow assumptions are met:

1. HTTP server is using HTTPS;
2. HTTP server has a valid SSL certificate;
3. HTTP server returns a valid Date header.

The first and third assumptions are met by most HTTP servers, and the high stake nature of blockchain applications implicitly preclude those incompatible servers. The second assumption can be met by integrating SSL certificate validation on-chain. With these assumptions met, we are able to construct a proof for an HTTP invocation history with the following elements:

1. HTTPS server certificate;
2. Diffie-Hellman parameters;
3. HTTPS server signatures;
4. Encrypted HTTP request and response.

The proof above allows us to validate an HTTP API is invoked at a specific timestamp because:

1. HTTPS server certificate certifies the server public key;
2. Diffie-Hellman parameters and server signatures compute the session key, which validates the encrypted HTTP data;
3. The Data header in HTTP response verifies the invocation timestamp.

Apsis EVM

Apsis is an HTTPS-empowered blockchain which supports retroactively verifiable HTTPS invocations. The application scope of Apsis can be maximized with Apsis EVM, an HTTPS-empowered smart contract platform which allows smart contracts to invoke HTTPS API synchronously and reliably.

The HTTPS access capability in Apsis EVM is provided with an EVM precompile. Below is a sample of the HTTP precompile written in Solidity.

```
contract HTTPClient {

// Format of the HTTP response content.
enum Format{ JSON, TEXT, BINARY };

// Reads data from an HTTP endpoint.
// @param url URL to invoke
// @param format Format of the HTTP response content
// @param path Path of the data to return
// @param expiration Deadline of the HTTP invocation
// @return The data specified as JSON path
function getData(string url, Format format, string path, uint256 expiration) public returns (bytes memory);

}
```

The code snippet below shows how to read asset price using HTTP and rebase in a single transaction:

function rebase() external { const _httpClient = HTTPClient.at('0x0000000000008'); const _data = httpClient.getData('https://api.token.com/prices/AMPL', HTTPClient.Format.JSON, 'price.value', 10 minutes); uint256 _price = abi.decode(_data, {uint256}); _rebaseWithPrice(_price); }

When HTTPClient precompile is called, in addition to reading data from HTTP API, it appends HTTPS session proof as part of the transaction log. To avoid the problem of bloating data storage, the proof data for relatively old blocks can be packaged and offloaded to external decentralized content-addressable storage such as IPFS.

Applications

One notable application of Apsis is On-chain Oracles which provides end-to-end verifiability to access external data.

[

image

902×547 29.8 KB

](https://ethresear.ch/uploads/default/original/2X/b/b30bb5dadeaeb0e22aa60844b5e989ade42f194e.png)

There are two critical differences between off-chain Oracles and on-chain Oracles:

- Oracle nodes are no longer required in on-chain Oracles since on-chain Oracles can access HTTP API directly. This might imply significant operational cost since no middleman tax is paid to Oracle nodes;
- Data from HTTP API to Oracle contract is completely verifiable in on-chain Oracles. This means the data consumers only need to trust the HTTP API providers, in contrast to off-chain Oracles where data consumers need to trust the Oracle nodes to operate honestly.

The presence of on-chain Oracle does not preclude the existence of third-party Oracle providers. Existing Oracle providers can build their Oracle solutions on Apsis which read data from multiple HTTP APIs and aggregate data points. There are several benefits to build on-chain Oracle solutions on Apsis:

- On-chain Oracles on Apsis provide end-to-end data verifiability. Oracle data consumers don't have to trust the Oracle providers as the whole data generation process is visible on-chain;
- Oracle providers don't have to manage their Oracle node networks and overpay them with their own tokens. Instead, Oracle providers can better utilize their token to bootstrap the ecosystem;
- On-chain Oracles can support both realtime and cached data access. Oracle data consumers can pay more to trigger a new round of data update in order to read the latest data, or pay a small portion to access the data cached in the last round update.

On-chain Oracles can also be accessed by applications on other blockchains via cross-chain bridge. For example, lending protocols on Ethereum can utilize the Apsis-Ethereum bridge to read the latest price from the on-chain Oracles on Apsis. Apsis chain can become an Oracle hub in the incoming multi-chain era.

Apsis can also empower hybrid blockchain applications to take full advantage of both Web 2.0 and 3.0. For example, an electronics retailer can build a sales DApp to sell their televisions and use their existing systems to provide price and inventory information. The whole purchase can be implemented in a single transaction which is not feasible on any existing blockchains.

For more information about Apsis, please refer toApsis' lightpaper.