Apsis: Web 2.0-Compatible Blockchain

Shengda Ding

1 Overview

Satoshi Nakamoto's Bitcoin introduces the concept of blockchain and signals the emergence of crypto currency. It effectively becomes the first blockchain application in the world. Ethereum expands the application scope of blockchain, extending from distributed payment network to generic computation platform for decentralized contracts and applications. Decentralized finance, mostly notably known as DeFi, is the dominant decentralized application on blockchain. Since Compound launched liquidity mining in August 2020, we have witnessed a more than 10 times growth in DeFi total locked value within 6 months.

The prosperity of DeFi makes two shortcomings of existing blockchain applications more apparent: low throughput and isolation from the external world. While the former can be alleviated by new technologies such as sharding and rollup, there is no easy solution to the latter since existing blockchains are inherently isolated networks. Therefore, current DeFi applications heavily rely on Oracle to access the external data, and Oracle effectively becomes the security bottleneck of blockchain applications since its performance cannot be effectively validated on-chain.

Apsis aims at solving the Oracle problem of the blockchain applications. By introducing verifiability and retroactivity to HTTP invocations, Apsis becomes the first Web 2.0-compatible blockchain which allows smart contracts to interact with HTTP API synchronously in a single transaction. Apsis can expand the application scope of blockchains with hybrid applications which take full advantage of both Web 2.0 and Web 3.0 worlds.

2 Oracle: Solutions and Problems

Since its inception, blockchain has been an isolated and self-contained computation platform. If data is not available on chain, it must be fed on chain in order to be processed by smart contracts. Any role that takes the data feeder responsibility is called Oracle.

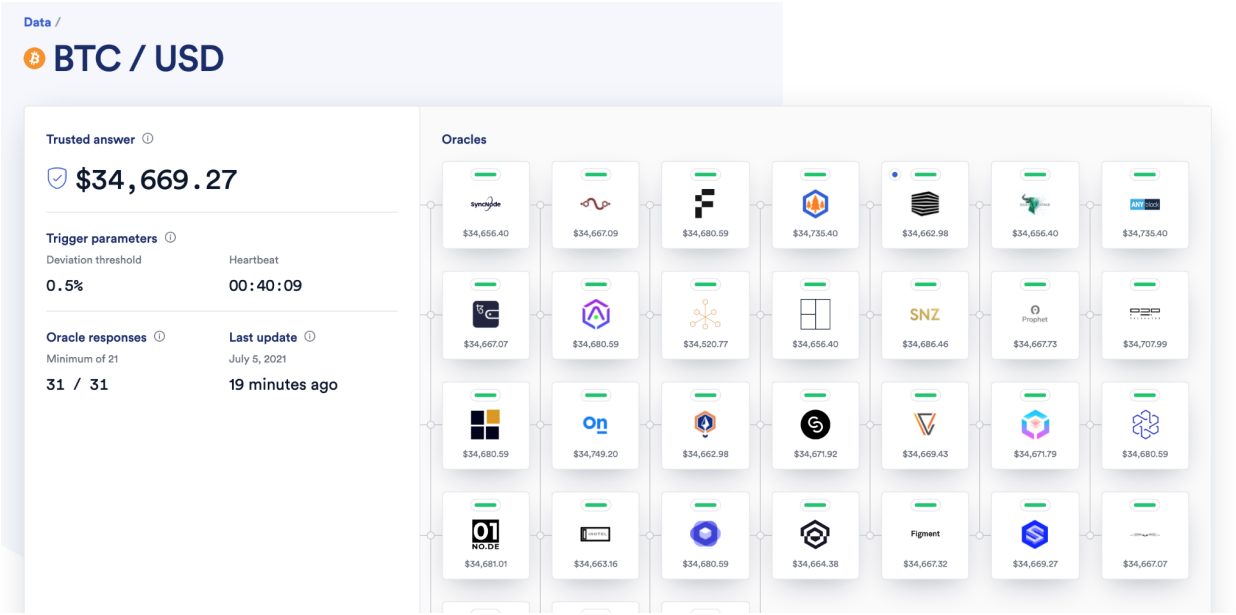
Block applications, including the emerging DeFi applications, heavily rely on off-chain data provided by Oracle to function properly. For example, lending and exchanges rely on Oracle to provide asset prices, and prediction markets rely on Oracle to provide certain results, e.g. the winner of 2020 US presidential election. However, existing Oracle solutions can only offer extremely limited security and reliability when compared to blockchain, which effectively makes Oracles the shortest stave in Liebig's barrel.

2.1 Existing Oracle Solutions

The purpose of this section is to go through major Oracle solutions in both Ethereum and Polkadot ecosystems. Note that Nest is not included since it's a custom solution for on-chain asset prices while we are looking at generic Oracle solutions.

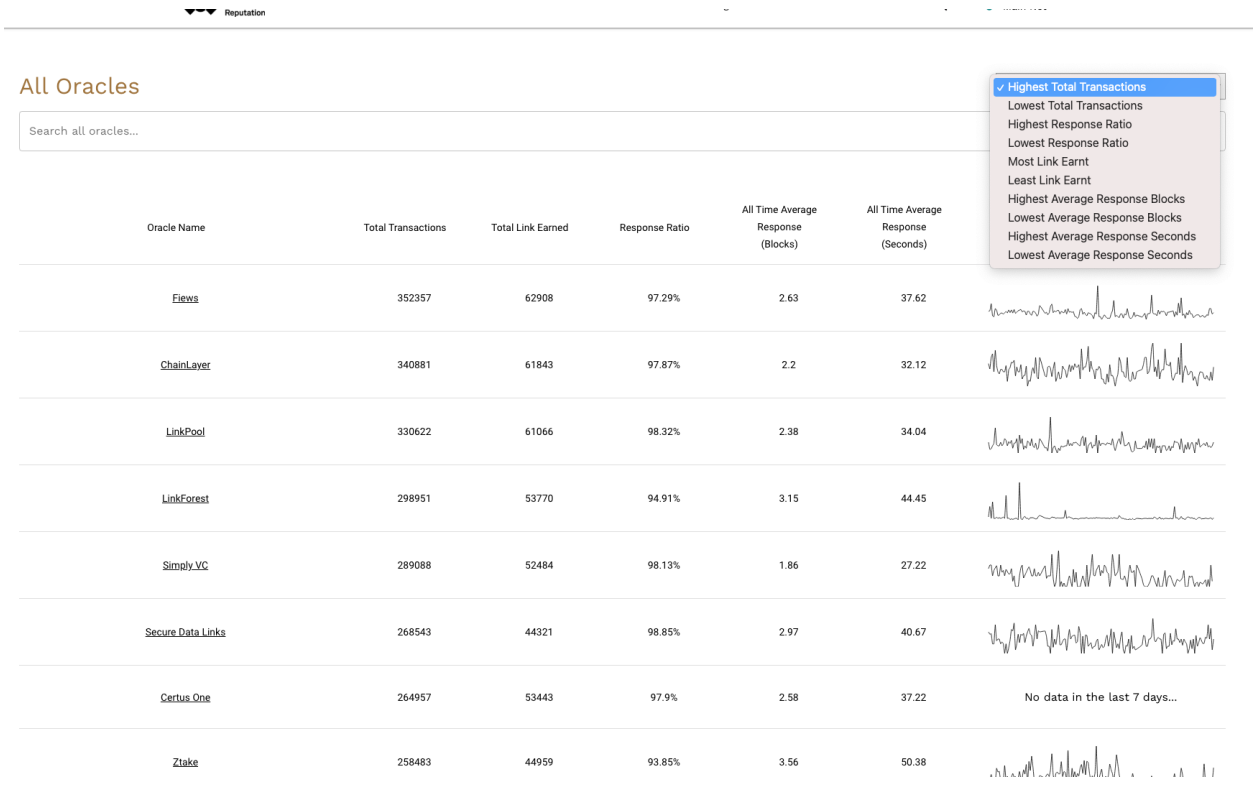
2.1.1 ChainLink

ChainLink, undoubtedly, is the largest and most widely used Oracle in DeFi. ChainLink centrally manages a list of trusted Oracle nodes which are operated by well-known companies. All Oracle nodes in the same data market, e.g. BTC/USD price, form a Decentralized Oracle network. Each Oracle node manages their own list of data sources, usually paid and premium ones, and aggregates the data using their own algorithm.



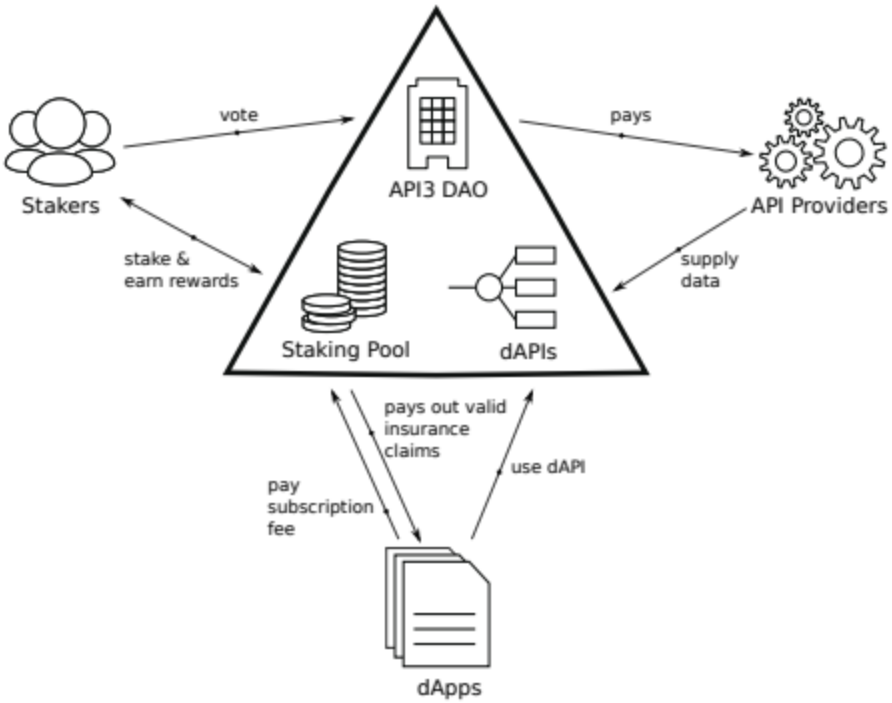
When the target price deviates from a certain threshold or the heartbeat timer expires, all nodes in the Decentralized Oracle Network report new data points which are aggregated and reported on-chain via a randomly chosen network leader.

ChainLink is effectively a centralized data marketplace. Even though the Oracle nodes form a decentralized network to aggregate price data, they are centrally managed by the ChainLink team. Moreover, detailed information about the Oracle nodes, such as the data sources used, are completely hidden to public users so they have no way to verify whether these nodes are reading, aggregating and reporting data honestly. ChainLink tries to solve the problem with off-chain reputation, but the existing reputation system is purely based on Oracle node responsiveness instead of data quality.



2.1.2 API3

API3, designed and implemented by a former ChainLink Oracle node operator, aims to solve the data opacity issues in ChainLink. They find that the API providers, which often have big off-chain reputations, are completely hidden in the ChainLink architecture. Therefore, instead of letting Oracle nodes earn the middlemen tax, they introduce Airnode, a first-party Oracle node which is operated by the API providers directly.

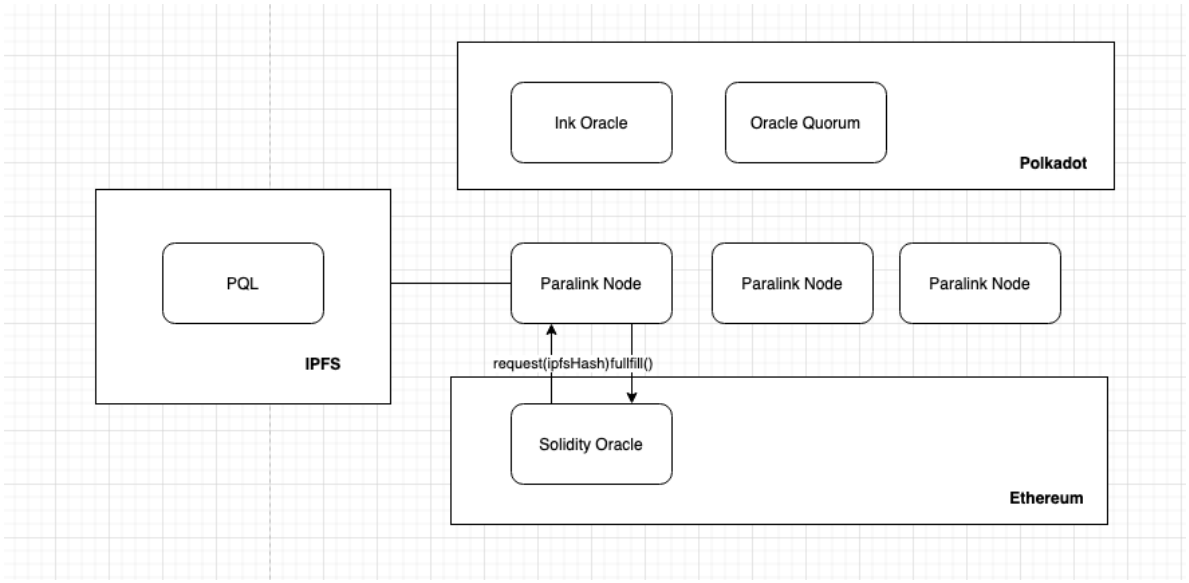


API3 makes the API providers visible to DApps. However, similar to ChainLink, DApp still has no way to verify whether the data really comes from the target API providers. API3 is also not easy to use. In order to access a new API provider, DApp developers must make a request to API3 DAO to create a decentralized API, or dAPI for short, for that API provider. DApp developers then should subscribe to the dAPI, create a subscriber address and fund it with sufficient ETH so that API providers can use it to provide data. In short, API3 is a custom solution targeted at the API providers, but it does not completely solve the data opacity issue in Oracle and introduces new usability issues.

2.1.3 Paralink

Paralink is an Oracle solution on Ethereum and Polkadot. Similar to ChainLink and API3, Paralink utilizes off-chain Oracle nodes, called Paralink nodes, to access external data. However, Paralink offers two additional improvements:

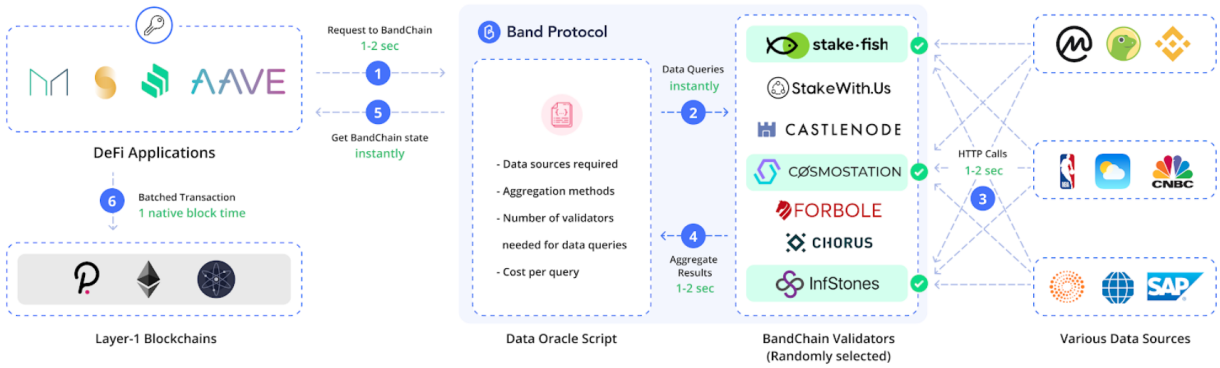
1. Paralink defines an Oracle script language, named Paralink Query Language, which enables flexible definition on data source and data aggregation operations;
2. Paralink offers on-chain Oracle nodes quorum on Polkadot so that the Oracle nodes management is more decentralized and transparent.



Still, Paralink Oracle users need to trust Paralink Node to perform their Oracle script honestly since the Oracle users cannot verify the outcome of Oracle scripts.

2.1.4 Band Protocol

Band Protocol builds a Cosmos-based blockchain, called Band Chain, to perform Oracle operations.



Similar to Paralink, Band Chain defines an Oracle script which allows Oracle users to specify data source, data aggregation and data validation operations. For each data source, Band Chain uses DPoS algorithm to randomly select multiple validators to perform the read operation. Data read by the validators are aggregated and their medium is reported as the final result.

Band Chain is one step ahead of the other Oracle solutions above since it creates a dedicated Oracle blockchain. This allows public users to verify the past output of the Oracle. For example, if an Ethereum transaction should consume data from the Band Chain, the public users are able to verify how the data is aggregated from various data points. However, users cannot verify

whether the data points come from the actual target data sources. There is no guarantee that validators of Band Chain do their right job since a medium aggregation approach cannot preclude collusion. For example, says the Band Chain uses 3 validators for each data source, if the attackers happen to control 2 of the select validators, they will be able to forge the data.

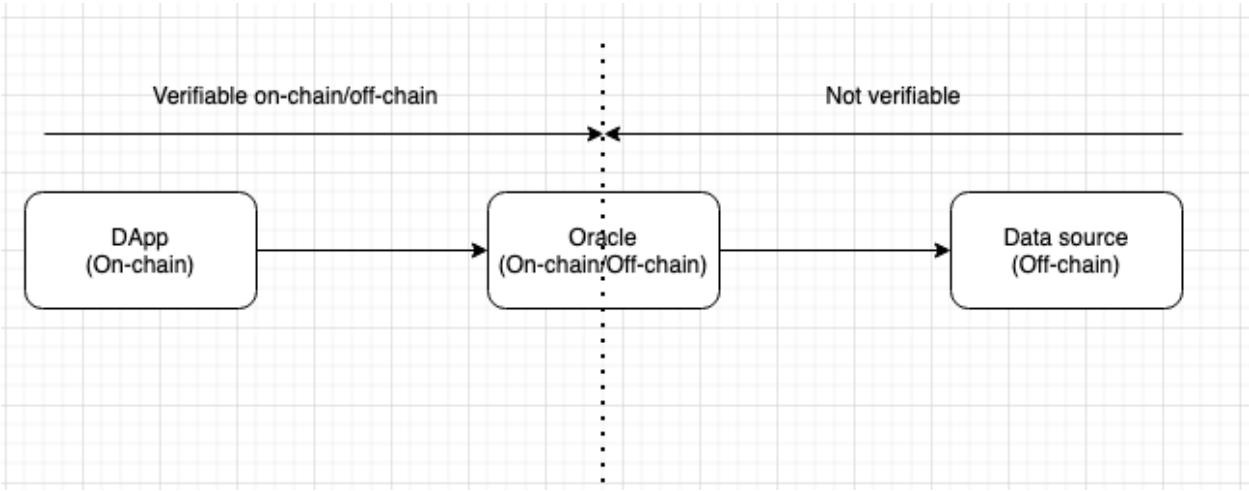
2.2 Problems of Oracles

Existing Oracle solutions fail to match the security demand of blockchain applications as they can't provide verifiability to their data. Even though they use token economics to alleviate the issues, this does not solve the fundamental problems mentioned below.

2.2.1 Lack of Off-chain Verifiability

The most critical issue of Oracles is that they cannot prove the validity of their data, i.e. their data is read from the targeted data source.

- Data sources are completely hidden in ChainLink. Public users don't know where their data come from, and ChainLink reputation system does not verify data sources;
- Even though data sources are more visible in API3 and Paralink, Oracle users still have to trust the Oracle nodes to provide the correct data;
- Band Protocol uses a quorum of validators to read from the same data source. However, the medium approach cannot preclude collusion and it does not work for all data types.



The data path from data source to DApp can be divided into two parts: The path from data source to Oracle, and the path from Oracle to DApp. While existing Oracle solutions try to prove the latter with either on-chain or off-chain proof, proof for the former is still in the air. We cannot verify the Oracle data by simply invoking the data source again, since its return data might change already. This is especially true for blockchain data which need to be validated long after the transaction is minted. Lack of off-chain verifiability between Oracle and data source make existing Oracle solutions unreliable and unsecure.

2.2.2 Heavy Reliance on Token Economy

Since off-chain data cannot be verified on-chain, existing Oracle solutions mostly rely on token economics to incentivize Oracle node operators to perform honestly. Token economy works for Bitcoin and Ethereum and bootstrap their security foundation.

However, things are different for Oracles. For Bitcoin and Ethereum, if any miners tries to forge a transaction or a block, the transaction/block will be excluded by honest miners. In such cases, the dishonest miner loses his reward right away. Oracle is more of a hindsight: the dishonest action is usually recognized only after the real damage is done. This is mostly due to the lack of off-chain verifiability to data source data, and token economy cannot solve the problem.

Therefore, Oracle usually needs to overpay their operators. Assume that all operators are rational. In order to incentivize honest behavior, the following conditions should always be met: For all situations, the reward for being honest, plus future income, should always be higher than the one-off income of being dishonest.

As suggested in the API3 whitepaper, the real life situation will be more complicated:

1. People tend to undervalue their future income and overvalue the one-off income of being dishonest due to the uncertainty of future;
2. The one-off income of being dishonest can be very large, considering the burst of DeFi total locked value. For example, the Oracle price data might determine whether a lending position should be liquidated or not. Oracle operators might choose to forge the data to gain the liquidation benefit.

In short, Oracle solutions have to overpay their operators to perform honestly in each interaction, which makes the Oracle pretty expensive. The Oracle might fail if the incentive is lower due to governance change or token price drops. The overpayment also does not preclude the free riders: Oracle node operators can simply forge data that falls into the expected range. They have no risk of losing future income by providing such low-quality data.

2.2.3 Pull vs Push Methods

DApp has two approaches to access external data via Oracle: pull and push methods.

- In the pull method, DApp triggers a data request event signaling the desired data to retrieve. Oracle nodes pick up the request, complete the job and send data back to DApp;
- In the push method, data is pushed to Oracle by nodes periodically or when a certain threshold is met.

The pros and cons of the pull and push methods are discussed in the table below.

	Pros	Cons	Example
Pull method	Data is up-to-date;	Asynchronous	ChainLink v1, Band

	support dynamic request URL	invocation, difficult for programing	Chain, API3, Paralink
Push method	Synchronous invocation, programming friendly	Data is stale; difficult to set proper push schedule; data URL must be known beforehand	Band Protocol v1, ChainLink v2

Note that even though the pull method tends to offer fresher data than the push method, the freshness is still dependent on the Oracle node. If the Oracle node does not pick up the request or respond in time, pull-based Oracle users still suffer from outdated data.

3 Apsis: The First Web 2.0-Compatible Blockchain

Oracle is an essential component of existing blockchain applications since the state-of-the-art blockchain is isolated from Web 2.0. Blockchain applications have to rely on an external actor to invoke Web 2.0 APIs and feed the data back. As Liebig's law of minimum suggests, weakness of Oracle effectively becomes weakness of the blockchain applications, and blockchain application developers have to trade security for external data accessibility.

Apsis chain can achieve security and external data accessibility at the same time. By supporting retroactively verifiable HTTP invocations, Apsis chain is the first Web 2.0 blockchain which could interact with existing HTTP APIs with no security downgrade. Apsis eliminates the dependency on external Oracles, and empowers hybrid blockchain applications which can take full advantage of the security of Web 3.0 and the diversity of Web 2.0.

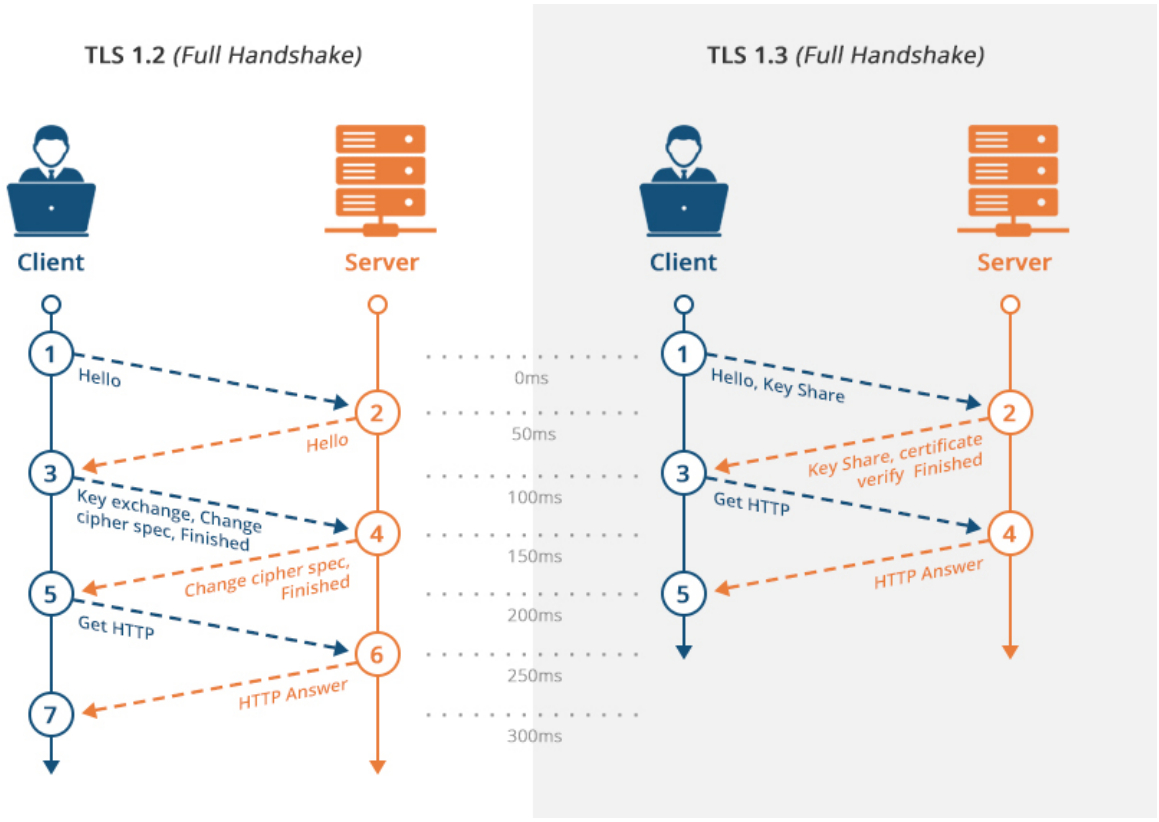
3.1 Retroactively Verifiable HTTP Invocations

All blockchain transaction data need to be retroactively verifiable. The validity of past transactions can be verified by any others, even when these transactions are included on-chain a long time before. HTTP, on the other hand, does not provide such native verifiability and retroactivity. We cannot verify an HTTP response by simply invoking the HTTP API by ourselves because the HTTP API data might change already, This is especially true for HTTP responses recorded long ago.

HTTP invocation can be retroactively verified as long as the following assumptions are met:

1. HTTP server is using HTTPS with TLS 1.3;
2. HTTP server has a valid SSL certificate;
3. HTTP server returns a valid Date header.

Even though TLS 1.2 can offer similar verifiability, we are dedicated to TLS 1.3 due to its forward security, efficiency and simplicity. Below is a comparison between the TLS 1.2 and TLS 1.3 handshake process.



If these three assumptions are met, we are able to construct a proof for an HTTP API invocation with the following elements:

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

HTTPS server certificate certifies the HTTPS server public key. Combined with Diffie-Hellman parameters and the server signatures used in the HTTPS session, we are able to generate and verify the share key used in this session, which is in turn used to verify the HTTP request and response content.

In other words, the first two assumptions, i.e. HTTPS with valid certificates, allow us to generate a proof that the data is read from a target server honestly. This makes the HTTP API invocation verifiable. The third assumption, i.e. a valid Date header in the HTTP response, makes this verification retroactive with a persistent proof for the HTTP response timing. All these combined generate a proof of whether and when an HTTP API is invoked so that it can work with other on-chain transaction data.

One might argue that not all Web 2.0 HTTP servers can meet these three assumptions. We will discuss how we handle SSL certificates in section 3.3. For assumption 1 and 3, fortunately, almost all HTTP servers nowadays work with HTTPS and they are able to return a standard

valid Date header. For those incompatible servers, they will be precluded implicitly by the blockchain applications. Since blockchain applications require gas to deploy smart contracts and execute transactions on-chain, they are more suitable for high-stake applications with high security demand. Therefore, these applications are more likely to choose HTTPS servers with similar security. It's thus highly unusual for them to select an API that does not support HTTPS or does not follow the HTTP standard.

In short, Apsis chain is not supposed to work with all Web 2.0 servers, but it should work with HTTP servers that are needed by blockchain applications. This does not change the fact that Apsis is the first blockchain that could interact with Web 2.0 servers.

3.2 Web 2.0-Compatible Smart Contract

Apsis chain is an EVM-based smart contract platform which enables smart contracts to invoke HTTP API synchronously in a single transaction. The HTTP invocation functionality 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);
}
```

Below is a sample method to invoke the HTTPClient precompile.

```
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);
}
```

The code snippet tries to read the latest price of AMPL token to determine whether a rebase operation is needed. Assume that the (faked) URL https://api.token.com/prices/AMPL returns a JSON which contains AMPL's latest price under path 'price.value'. When the getData method is invoked on the HTTPClient precompile, it invokes https://api.token.com/prices/AMPL, reads and parses the JSON response, and returns the bytes data as result. Miners will append proof of the HTTPS invocation, as specified in section 3.1, as part of the transaction log.

One potential problem is the storage size. As transactions involving HTTP invocation accumulate, HTTPS session proof is likely to bloat and consume large amounts of storage space. This problem can be addressed by offloading HTTPS session proofs to external decentralized storage. For example, for relatively old blocks, we could package HTTPS session proofs for every 100 blocks and store them in other decentralized, content-addressable storage solutions such as IPFS, while leaving the response data(e.g. the return value of getData() in the example above) stored on-chain. Therefore, if miners want to validate relatively old transactions, they can download the HTTPS session data and discard them afterwards.

Note that the HTTPS invocation is synchronous as part of the transaction. It provides better development experience compared to the pull-based Oracle, and it allows ad-hoc HTTP API access which is difficult to achieve in push-based Oracles. In addition, it offers fresher data than both push- and pull-based Oracle solutions as the data is read in the same transaction.

3.3 SSL Certificate Integration

As mentioned in section 3.1, the second assumption to meet is that HTTPS server should have a valid SSL certificate. To verify an HTTPS invocation record, Apsis miners need to validate its SSL certificate at the same time of the HTTPS invocation, since this SSL certificate might become invalid at the time of validation.

To help achieve global consensus on SSL certificate validity, Apsis integrates SSL certificate validation as part of the blockchain implementation. Apsis maintains both the root CA certificates and the Certificate Revocation Lists(CRL) on-chain. Whenever new certificates are revoked, Apsis DAO, which is discussed in detail in Section 3.4, submits a change to CRL so that the newly revoked certificates, along with the revocation date, are updated on-chain. The process to validate an SSL certificate at a specific timestamp is as follows:

1. Check whether the certificate has valid proof from CA. If no, the certificate is invalid;
2. Check whether the certificate expires before the target timestamp. If yes, the certificate is invalid;
3. Check whether the certificate is on the CRL. If so, check whether its revocation timestamp is before the target timestamp. If yes, the certificate is invalid;
4. Otherwise, the certificate is valid at the target timestamp.

Since anyone can validate an SSL certificate for any given timestamp, Apsis chain can be seen as a public welfare chain on its own, since it could solve the availability and usability issues of the current Online Certificate Status Protocol(OCSP).

3.4 Token Economics and Apsis DAO

APS is the native token of the Apsis chain. Users need to consume APS as the gas fee to deploy smart contracts and execute transactions. Similar to ETH, APS is used to cover the computation and storage cost at Apsis chain, especially the additional cost of HTTPS invocation and HTTPS session proof storage.

APS holders can stake APS to become, or delegate others to become validators of the Apsis chain. Apsis chain adopts Delegated Proof-of-Stake (DPoS) as its underlying consensus protocol. Proof-of-Work(PoW) is definitely not an option since Apsis transactions might interact with HTTP servers. If PoW is adopted, multiple miners might be executing the same transaction with HTTP invocation, which effectively becomes a DDoS attack to the HTTP server.

APS stakers, no matter whether they delegate themselves or others, will become members of Apsis DAO to participate in Apsis chain governance. Apsis DAO is responsible for the following works:

- Empower Apsis chain development;
- Bootstrap and incentivize Apsis ecosystem building;
- Perform routine chain governance operations, such as update the latest Certificate Revocation List on-chain.

3.5 Apsis Applications

With the incoming multi-chain era, Apsis can serve as a Web 2.0-compatible blockchain on its own or provide Web 2.0 accessibility to other chains via bridge.

3.5.1 Oracle Hub

Apsis chain makes the interaction with HTTP servers transparent and retroactively verifiable. It solves the data verification problem between DApp and data sources so that existing Oracle solutions can be enhanced with data provider contracts on the Apsis chain. These data provider contracts can invoke multiple HTTP APIs to generate an aggregated result. The HTTP server interactions and the data aggregation process can be verified by any public users. Another benefit of Apsis data contracts is that it does not rely on token economy to insure off-chain operations. Existing Oracle solutions can better utilize their tokens instead of overpaying Oracle node operators and hoping that they will always perform honestly.

When data provider contracts are available on the Apsis chain, DApp on other chains can also access its data via bridges. For example, the lending and derivative protocols on Ethereum use data provider contracts on Apsis to provide realtime asset prices. This can be done via an Apsis-Ethereum bridge. Apsis can also be developed and deployed as a Polkadot parachain, where other parachains can access external data via Polkadot's Cross-Chain Message Passing protocol.

In short, Apsis can become the Oracle hub for all other blockchains.

3.5.2 Web 2.0 Enhancement

Apsis chain is also an enhancement to Web 2.0. Since blockchain applications are expensive to execute, they are mostly high stake applications with high security and reliability demand. It is not a surprise if they expect the same security and reliability on their dependent HTTP APIs.

In order to track the performance of HTTP APIs, traditional Web applications need to build their own tracing solutions, recording individual responses from the API providers, and providing proof if they believe their API providers do not perform as expected.

With Apsis, all responses from API providers are visible on-chain and can be verified by any public users. If any DApp incidents happen, it is not difficult to determine whether the issue comes from the dependent HTTP APIs or the blockchain application itself. For example, if one's lending position is liquidated due to low loan-to-value ratio, we could figure out whether it's caused by asset price change or smart contract flaw. If it's caused by asset price change, say, a drop in collateral price, we are able to figure out how the price is read and calculated, eventually finding out what data sources drive the price shift. The whole liquidation process is transparent.

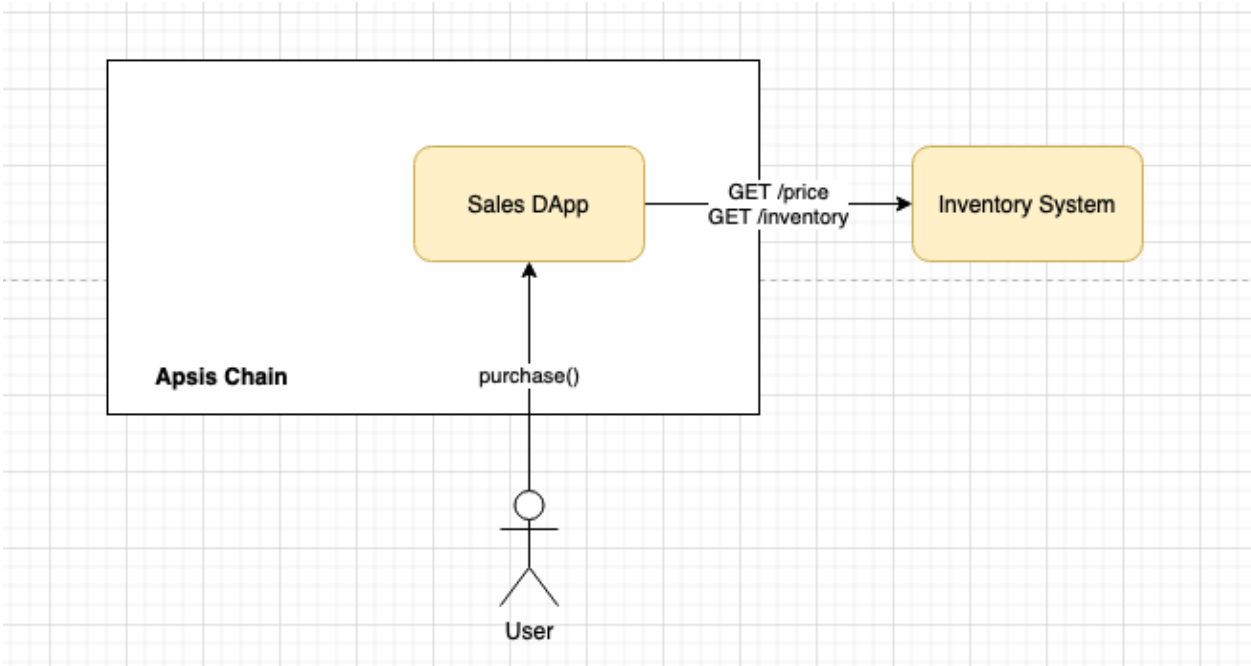
In short, Apsis brings retroactive verifiability to Web 2.0 APIs. One could build a reputation system based on the HTTP API provider performance since their past responses are recorded on-chain and can be verified by any users.

3.5.3 Hybrid Blockchain Applications

The true potential of the Apsis chain is definitely not limited to Oracle. Since smart contracts on Apsis can access HTTP API synchronously within a single transaction, developers are able to write smart contracts involving complicated HTTP interactions, which are currently not available in any existing blockchain.

In other words, Apsis opens the space for totally new kinds of applications, which we call hybrid blockchain applications, to take full advantage of both the security and reliability from Web 3.0, and diversity and complexity from Web 2.0.

Assume that an electronics retailer wants to build a blockchain application to sell their televisions. Since there are many TVs available and their inventory information changes frequently, the retailer does not want to maintain all information on-chain. Instead, they rely on their in-house inventory system to provide price and inventory information. Users can purchase TV with cryptocurrency via the sales blockchain application and anyone can verify the sale transaction information on-chain.



It is difficult to implement the sales DApp using existing Oracle solutions. Push-based Oracle doesn't work since it will be expensive to push all prices and inventory information on the chain. Pull-based Oracle also has its own issues.

- First, the purchase experience will have to break into multiple transactions since the pull-based Oracle is asynchronous. The problem is worse as the purchase involves multiple HTTP interactions, especially when these HTTP interactions depend on each other. In such a case, the users might have to sign multiple transactions with long time waiting to complete their purchase;
- Second, both users and the retailer have to trust the Oracle to return the correct information from the inventory system. If something goes here, say, a TV is sold at $10, it would be difficult to figure out where the mistakes come from.

All these problems can be solved with Apsis. Since Apsis enables smart contract transactions with synchronous HTTPS invocations, the sale DApp can complete the purchase experience in a single transaction. In addition, if a TV is sold at $10, we could easily identify whether the $10 price comes from the inventory system since all HTTPS interactions are recorded and can be verified by any users. Moreover, by eliminating the Oracle nodes as the middleman, the development and operational cost can be substantially smaller.

4 Security Analysis

The end-to-end security of Apsis transactions depends on both Apsis chain itself and the HTTP server. If we treat the HTTP server as a blackbox, we can show that:

1. When the HTTP server is secure, no one can forge transactions or invalidate a valid translation on Apsis;

2. When the HTTP server is breached, Apsis can provide a security guarantee no worse than the existing Oracle solutions.

The former is easy to prove. If one is able to forge an Apsis transaction which consists of at least one HTTPS invocation, they need to forge an SSL session shared key. To do that, they have to either obtain the HTTPS server private key or force the HTTPS server to use their forged SSL certificates. This requires breaking the HTTPS server or the encryption algorithm. Therefore, Apsis can offer much higher security and reliability guarantee than existing Oracle solutions if the HTTPS server is secure.

Even though the security of HTTP server is external to Apsis' security model, Apsis is no worse than the existing Oracle solutions when the HTTPS server is breached. We will discuss the detailed cases in the following sections.

4.1 HTTP Server Certificate is Breached

If the SSL certificate of the HTTP server is breached, the attacker can forge an Apsis transaction by creating a fake HTTPS response as proof. Likewise, in the existing Oracle solutions, the attacker can easily launch a man-in-the-middle attack against Oracle nodes. In a word, the HTTP server is no longer secure for either Apsis or Oracles.

Assume that the certificate breach is unknown to HTTP server operators. Neither Apsis nor Oracles can detect the breach, but Apsis can facilitate the discovery since the forged HTTPS sessions are recorded on-chain which allows HTTP server operators to verify the past sessions.

Assume that the certificate breach is known and thus added to RCL. How Oracle detects revoked SSL certificates is implementation specific. On the other hand, since Apsis is governed and operated by Apsis DAO, any Apsis DAO members can propose an RCL update on the chain. The decentralized nature of Apsis can reduce the damage time caused by the server certificate breach.

4.2 HTTP Server DNS is Hijacked

When the HTTPS server DNS is hijacked, the attackers still need to create a falsified SSL certificate to launch an attack. This is similar to what is discussed in section 4.1, and Apsis can help to detect falsified SSL certificates and react promptly.

4.3 HTTP Server is Breached

If the HTTP server itself is breached, neither Apsis miners nor Oracle nodes could do anything about it since the attacker can forge any response. However, Apsis can facilitate the breach discovery and investigation process. With all API responses recorded on-chain, users can identify the data coming from the HTTP server and locate the root causes in the HTTP server.

4.4 HTTP Server Returns Different Contents

HTTP servers might return different contents to different callers, even when they are invoking the same URL at the same time. For example, an HTTP response might be customized based on the caller's location or language. Even though it causes confusion to Apsis miners, we don’t consider it as error cases due to the following reasons:

1. If the HTTP server does not return a deterministic result, then this HTTP server might not be a good candidate for blockchain applications which requires high security and reliability;
2. Oracle might be able to alleviate the issues by aggregating results from multiple nodes, but it is not a generic approach and it can only provide an estimate for the data.

Therefore, if one Apsis miner receives a valid HTTP response and uses it to construct a valid block, it is considered as a valid transaction execution process.