XYO Network: Network Implementation

Erik Saberski, Carter Harrison, Arie Trouw

August, 2018

The XY Oracle Network utilizes a novel blockchain protocol to provide a trustless, cryptographic network of decentralized location data. The central mechanisms of the protocol are known as Proof of Origin and Bound Witness, which tie together blockchain technology and real-world data to make a system that has many direct applications today.

The XY Oracle Network is composed of four main components: sentinels, bridges, archivists, and diviners. Sentinels are offline IoT geolocation miners that communicate with each other to create a map of interactions. Bridges relay information from sentinels to archivists, which store the data. Lastly, diviners access the data on the archivists to answer queries made by end users. Each component has their own complexities, and in this paper, we explore exactly how each component works. We highly recommended for our readers to first read the XYO White Paper, which gives an overview of Proof of Origin, Bound Witness, and all other network components. Many of the details of this paper are built upon the ideas and concepts laid out there.

1 Network Encoding

In this section, we outline the protocol for sharing binary data between components in the XYO Network. The principles detailed here are utilized throughout the protocol.

To start, all binary data follows Big Endian. Also, sizes follow common semantic where size includes itself:

Bits	**Name**
8	Byte
16	Short
32	Int
64	Long

It is important to have an efficient way of encoding objects in the XYO Network given the limited space and computational power of IoT devices. To solve this, we utilize a type based encoding system where all object interpretation knowledge is held outside of the object itself.

1.1 Types

Types act as identifiers for certain objects and interpretations for binary data. A type may infer size, unpacking method, or any other relevant information. Types follow a major and minor structure for organization where the major is a group of objects and a minor is the field that describes the object type. All standard major and minor identification keys can be found in the appendix.

1.2 Structure

Objects are used for packing (and unpacking) data so the payload can be unpacked only knowing the Type of the payload. Thus, data will be organized in the following way:

Name	**Type**
Major Type	Byte
Minor Type	Byte
Value	Array <Byte>

A typed based binary system allows for modification without impact on existing protocols. Each structure defined is permanent (unless deprecated), meaning that any modification to a type will result in a new major and minor key pair. This allows for 2^{16} possible types and methods. A typed system has binary interpretation methods stored on the client-side, and 2 bytes (one for major and one for minor) referring to the interpretation method to use.

2 Bound Witness

A bound witness allows two parties to trustlessly agree on data using public key cryptography. With this, a third party can verify that the two parties came to an agreement after the bound witness is completed.

Bound witness works through a series of three transactions between two participating parties. Consider two parties: a client and a server. The client first sends their public key, along with any relevant data to the server. The server will add their public key, any data they want to share as well, and sign the public keys some of the data. The server will then send this back to the client, who will sign the same parts of the data and send it back to the server. This format is detailed below.

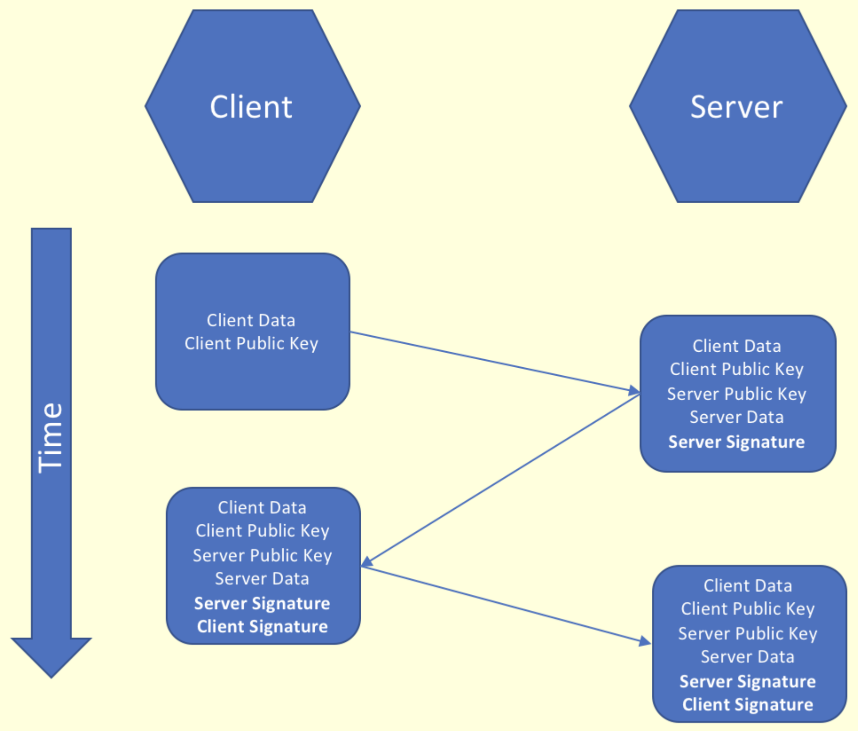

Figure 1: An example of a bound witness interaction. Each arrow represents an exchange of data between client and server, and the boxes represent what each holds after each exchange as the three transactions happen over time.

2.1 Bound Witness Block

Below is a detailed description of the parts of the data that are passed between client and server in a bound witness interaction.

Name	**Type**	**Included in Signature**
Total Size	Unsigned Integer	No
Public Keys	Array(0x02)<Keyset>	Yes
Payloads	Array(0x03)<Payload>	Yes and No
Signatures	Array(0x02)<Signature Set>	No

Note: Array types and other objects can be found in the appendix.

2.2 Payload

Note that above, payloads are both signed and not signed. That is because there are two arrays within the payload: a signed payload, and an unsigned payload.

Name	Type	Included in Signature
Total Size	Unsigned Integer	No
Signed Payload	Array(0x06)	Yes
Unsigned Payload	Array(0x06)	No

2.3 Interaction

Below shows a detailed interaction of two parties performing a bound witness interaction. Each stage shows what data is being sent, and what each party currently holds during each step (before data is sent). *Note: All data transfer has a header of the bound witness major and minor.*

2.3.1 Stage One

To initiate a bound witness the client sends the server the following:

Name	Type	Description
Total Size	Unsigned Integer	Size
Public Keys	Keyset	Client Keys
Payload	Payload	Client Payload

Client state:

Name	Type	Description	Final
Total Size	Unsigned Integer	Size	No
Public Keys	Array(0x02)<Keyset>	[Client Keyset]	No
Payloads	Array(0x03)<Payload>	[Client Payload]	No
Signatures	Array(0x02)<SignatureSet>	[]	No

Server state:

Name	Type	Description	Final
Total Size	Unsigned Integer	Size	No
Public Keys	Array(0x02)<Keyset>	[Server Keyset]	No
Payloads	Array(0x03)<Payload>	[Server Payload]	No
Signatures	Array(0x02)<SignatureSet>	[]	No

2.3.2 Stage Two

When the server receives the data they will add the client's payload and public keys to his bound witness copy, sign it, then sends the client the following:

Name	**Type**	**Description**
Total Size	Unsigned Integer	Size
Public Keys	Keyset	Server Keys
Payload	Payload	Server Payload
Signatures	SignatureSet	Server Signatures

Client state:

Name	**Type**	**Description**	**Final**
Total Size	Unsigned Integer	Size	No
Public Keys	Array(0x02)<Keyset>	[Client Keyset]	No
Payloads	Array(0x03)<Payload>	[Client Payload]	No
Signatures	Array(0x02)<SignatureSet>	[]	No

Server state:

Name	**Type**	**Description**	**Final**
Total Size	Unsigned Integer	Size	No
Public Keys	Array(0x02)<Keyset>	[Client Keyset, Server Keyset]	Yes
Payloads	Array(0x03)<Payload>	[Client Payload, Server Payload]	Yes
Signatures	Array(0x02)<SignatureSet>	[Server SignatureSet]	No

2.3.3 Stage Three

When the client receives the data they will add the server's payload, signatures, and public keys to their bound witness copy, sign it, then send the server the following:

Name	**Type**	**Description**
Total Size	Unsigned Integer	Size
Signatures	Keyset	Client Signatures

Client state:

Name	**Type**	**Description**	**Final**
Total Size	Unsigned Integer	Size	Yes
Public Keys	Array(0x02)<Keyset>	[Client Keyset, Server Keyset]	Yes
Payloads	Array(0x03)<Payload>	[Client Payload, Server Payload]	Yes
Signatures	Array(0x02)<SignatureSet>	[Client SignatureSet, Server SignatureSet]	Yes

Server state:

Name	**Type**	**Description**	**Final**
Total Size	Unsigned Integer	Size	No
Public Keys	Array(0x02)<Keyset>	[Client Keyset, Server Keyset]	Yes
Payloads	Array(0x03)<Payload>	[Client Payload, Server Payload]	Yes
Signatures	Array(0x02)<SignatureSet>	[Server SignatureSet]	No

2.3.4 Final Product

To complete the bound witness, the server will add the client's signatures. This gives us the final product:

Name	**Type**	**Description**	**Final**
Total Size	Unsigned Integer	Size	Yes
Public Keys	Array(0x02)<Keyset>	[Client Keyset, Server Keyset]	Yes
Payloads	Array(0x03)<Payload>	[Client Payload, Server Payload]	Yes
Signatures	Array(0x02)<SignatureSet>	[Client SignatureSet, Server SignatureSet]	Yes

2.4 Higher Order Bound Witness

Due to the XYO Network's specific utilization of bound witness with only two participants, and for simplicity's sake, we have only discussed bound witness interactions involving two parties in this paper. However, it is possible to have a bound witness interaction with more than two participants where multiple participants act as both a client and server.

3 Proof of Origin

All valid bound witness in the XYO Network will be in the form of an origin block. An origin block is a bound witness that includes all of the origin fields (described below) in the signed payload. Origin fields allow bound witnesses to be linked together so that an observer can check the validity of the chain post-hoc. A collection of origin blocks that can be linked together through the origin fields is called an Origin Chain.

3.1 Origin Fields

The four origin fields are index, previous hash, next public key, and hash of the heuristic data (the unsigned payload).

3.1.1 Index

The index is the number of an origin block in an origin chain. Meaning that the genesis block of an origin chain would have an entry of 0, and the first block would have an entry of 1, etc. An index is encoded as a 32 bit unsigned integer.

3.1.2 Previous Hash

The previous hash is a hash of the entire previous block in a given component's origin chain. Including the previous hash in every origin block turns an origin chain into a self-maintained blockchain. Unlike other blockchains, the hashing algorithm used is left up to party creating the origin chain. A more secure hashing algorithm is more likely to be considered as valid data when used to answer a query by diviners. Details regarding hash encoding can be found in the appendix.

3.1.3 Next Public Key (Optional)

Due to the insecurity of IoT devices, it is sometimes needed to rotate key pairs. When keys are about to be rotated, the next public key will be included in this field in any public key format.

3.2 Origin Chain

Each network component maintains an origin chain. This chain is compromised of origin blocks. Each time a component in the XYO Network goes through a bound witness interaction to produce an origin block, they store the signed information (unsigned is optional) in their origin chain. Because each origin block contains a the index, next public key used (if rotating keys), and the hash of the previous block in their chain, these chains can be followed both forward and backward.

Proof of Origin is the verifiable fact that each block in an origin chain is valid. If each block is shown to be a valid origin block within an origin chain, it is known that each of the interactions represented in the origin chain actually happened. For more information regarding Proof of Origin, please refrence the XYO White Paper.

3.3 Self Signing

Network components can complete a bound witness interaction with themselves if they wish to show activity in their chain when there are no other components to interact with. To do this, they simply include the origin heuristics as normal, and sign the data themselves.

4 Sentinels

Sentinels act as location witnesses within the XYO Network. Sentinels interact with each other via Bluetooth connection (with other means of connection in the future) to complete a bound witness interaction where they exchange heuristic data. Over time, their origin chains maintain a history of all other sentinels and bridges they have interacted with.

4.1 Sentinel to Sentinel Interaction

When two sentinels interact, one acts as the client and one acts as the server. They perform a standard bound witness interaction, including origin information (index, next public key, previous hash, hash of heuristics) in their signed payload, and heuristic data in their unsigned payload. Sentinels are strongly incentivized to keep the unsigned payloads in their origin blocks until they have been sufficiently offloaded to one or more bridges. Data offloaded without the unsigned payload (the heuristic data) will not be valuable in most cases, and it will be unlikely that they will be rewarded for offloading this data.

5 Bridges

Bridges collect data from sentinels and relay it to archivists. They have bound witness interactions with both the sentinels whom they receive data from, and the archivists who they relay data to. Bridges thus maintain an origin chain containing both sentinel-bridge and bridge-archivist interactions. These are described in the following sections.

5.1 Sentinel to Bridge Interaction

Bridges are constantly trying to make connections to sentinels. This connection is specific to the transport protocol. As such, they are the client in the bound witness interaction and the sentinel is the server. Once a bridge connects with a sentinel, they begin a bound witness interaction with the bridge sending a first packet of data with any optional heuristics in the unsigned payload, and any origin heuristics in the signed payload. The sentinel then adds their origin chain offload to the unsigned payload and their origin heuristics to the signed payload. The rest of the bound witness interaction continues as previously described. The steps are further detailed below.

1. The bridge initiates a bound witness interaction with a sentinel including their origin heuristics in the signed payload.

2. The sentinel adds their origin blocks they wish to offload into the unsigned payload, and origin blocks in the signed payload. This is detailed below:

Name	**Type**	**Description**
Total Size	Unsigned Integer	Total Size of Payloads
Signed Payload	Array(0x06)	Origin Heuristics
Unsigned Payload	Array(0x06)	Origin Blocks to offload

3. The bridge extracts the data from the unsigned payload of the origin block, and both the sentinel and bridge keep signed proof of the interaction in their origin chain.

5.2 Relaying Data to Archivists

As bridges extract data from sentinels' origin blocks as described in the previous section, they store it temporarily to transmit it to archivists. They connect to archivists through a bootstrapping node. Once they have offloaded the data to one or multiple archivists, they can delete the data. However, they must still keep the origin block from the transaction with the sentinel in their origin chain.

When bridges send data to an archivist, they have a bound witness interaction as well. Once the bridge sends the data to an archivist, the archivist initiates the bound witness interaction, thus making the archivist the client. The steps are described as follows:

1. The archivist receives the connections request. If they accept, the archivist receives data from the bridge.

2. The archivist extracts the payload hash of each piece of data they received and puts these in the unsigned payload in a bound witness interaction with the server archivist. This is specified below.

Name	**Type**	**Description**
Total Size	Unsigned Integer	Total Size of Payloads
Signed Payload	Array(0x06)	Origin Heuristics
Unsigned Payload	Array(0x06)	Hashes Within Origin Blocks

3. The bridge receives this bound witness interaction request, and the two complete the interaction.

4. Both the bridge and archivist store the block in their origin chain. However, the archivist must keep the unsigned hashes of the data in his origin chain (i.e. they must leave the unsigned payload) so that each piece of data can be traced back to the bridge.

6 Archivist

It is the archivists' job to store, share, and validate data that is collected by sentinels. They are rewarded for completing these tasks well and are punished for failing to do so. In order to maintain efficiency in our network, their goals are to ensure that all data on the archivist network is valid and spread all data as widely as possible. The more archivists that hold a given piece of data, the more quickly and more likely it will be found by a diviner to answer a query.

6.1 Sharing Data

All archivists' data is publicly available. However, in order to take data, an archivist must complete a bound witness interaction with the archivist they are taking data from. An archivist that is taking data is the client in this interaction, and the archivist providing the data is the server. When an archivist takes data from another archivist, they initiate a standard bound witness interaction including the payload hash of each piece of data they took in the unsigned payload of this interaction. The specific steps are detailed below.

1. The client archivist takes any data they want from the server archivist.

2. The client archivist extracts the payload hash of each piece of data they took, and puts this in the unsigned payload in a bound witness interaction with the server archivist. This is specified below.

Name	**Type**	**Description**
Total Size	Unsigned Integer	Total Size of Payloads
Signed Payload	Array(0x06)	Origin Heuristics
Unsigned Payload	Array(0x06)	Hashes Within Origin Blocks

3. The server receives this bound witness interaction request, and the two complete the interaction.

4. Both the client and server archivists store the block in their origin chain. However, the client archivist must keep the unsigned hashes of the data in his origin chain (i.e. they must leave the unsigned payload) so that each piece of data can be traced back to the server archivist.

Notice that the client archivist must keep the unsigned payload in his origin block. This is because when an individual piece of data is used by a diviner, the archivist must be able to prove where they got the data from. If the archivist does not keep the unsigned payload, all they have in their signed payload is a hash of all of the data they took. With only this hash, it is impossible to verify where an archivist got a specific piece of data unless they provide all of the data they took.

The server archivist, however, does not need to keep the unsigned payload. This is because his hash in the signed payload can be verified by the maintained unsigned payload in the client archivist's origin chain.

This could also be solved by simply putting the hashes of each piece of data in the signed payload. However, this is data that is unnecessary for the server archivist to store, and thus inefficient.

7 Diviner

It is the diviner's job to answer queries proposed on the network by finding relevant data stored in archivists. Many diviners can work together to answer a given query. In this case, as the diviners locate relevant data in archivists, they share it among each other, and come to consensus on what the best answer is.

7.1 Archivist to Diviner Interactions

A diviner can search through archivists to take any data they want that is relevant to answering a given query. However, in order for their data to be proven as valid for the given query, they must prove where they got the data from through a bound witness interaction with the archivist they took the data from. This bound witness interaction is exactly the same as the bound witness interaction between two archivists, where the diviner in this case is the client.

Appendix

Arrays (0x01)

Single Element Array

A weak array is used when there is an array with different types (e.g. heuristics).
Byte Size Array Header (0x01)

Name	Type	Description
Total Size	Unsigned Byte	Size of Entire Array
Type Major	Byte	Type Major of the element
Type Minor	Byte	Type Minor of the element

Short Size Array Header (0x02)

Name	Type	Description
Total Size	Unsigned Short	Size of Entire Array
Type Major	Byte	Type Major of the element
Type Minor	Byte	Type Minor of the element

Int Size Array Header (0x03)

Name	Type	Description
Total Size	Unsigned Int	Size of Entire Array
Type Major	Byte	Type Major of the element
Type Minor	Byte	Type Minor of the element

Array Element

Name	Type	Description
Element	Indicated in Header	An element in the array

Multi Element Array

A Multi Element Array is used when there is an array with the same type (e.g. an origin chain). **Byte Size Array Header (0x04)**

Name	Type	Description
Total Size	Unsigned Byte	Size of Entire Array

Short Size Array Header (0x05)

Name	Type	Description
Total Size	Unsigned Short	Size of Entire Array

Int Size Array Header (0x06)

Name	Type	Description
Total Size	Unsigned Int	Size of Entire Array

Array Element

Name	Type	Description
Type Major	Byte	Type Major of the element
Type Minor	Byte	Type Minor of the element
Element	Indicated Above	An element in the array

Core Objects (0x02)

Bound Witness (0x01)

A bound witness in the XYO Network.

Name	Type	Description
Total Size	Unsigned Int	Size of Entire Bound Witness
Public Keys	Array(0x02)<KeySet>	A keyset for every party in the bound witness.
Payloads	Array(0x03)<Payload>	A payload for every party in the bound witness.
Signatures	Array(0x02)<SignatureSet>	A SignatureSet for every party in the bound witness.

KeySet (0x02)

A set of public keys for a single party.

Name	Type	Description
Keys	Array(0x05)	A multi-typed array of public keys.

SignatureSet (0x03)

A set of signatures for a single party.

Name	Type	Description
Signatures	Array(0x05)	A untyped array of signatures.

Payload (0x04)

A payload contains a set of unsigned and signed Heuristics.

Name	Type	Description
Total Size	Unsigned Int	Total size of Payload
Signed Heuristics	Array(0x06)	An multi-typed array of heuristics
Unsigned Heuristics	Array(0x06)	An multi-typed array of heuristics

Index (0x05)

The index in an origin block.

Name	Type	Description
Index	Unsigned integer	the index in the origin chain

Previous Hash (0x06)

The previous hash in an origin chain.

Name	Type	Description
Hash Type Major	Byte	Hash Type
Hash Type Minor	Byte	Hash Type
Hash	Inferred above	the previous hash

Next Public Key (0x07)

The next public key to use in an origin chain.

Name	Type	Description
Public Key Type Major	Byte	Public Key Type
Public Key Type Minor	Byte	Public Key Type
Next Public Key	Inferred above	The next public key in an origin chain.

Hashes (0x03)

MD2 (0x01)

Calibration: MD2(0x010203)=0x30BD026F5B88B4719B563BDDB68917BE

Name	Size	Description
Hash	16 Bytes	MD2 Hash

MD5 (0x02)

Calibration: MD5(0x010203)=0x5289DF737DF57326FCDD22597AFB1FAC

Name	Size	Description
Hash	16 Bytes	MD5 Hash

SHA1 (0x03)

Calibration: SHA1(0x010203)=0x7037807198C22A7D2B0807371D763779A8
4FDFCF

Name	Size	Description
Hash	20 Bytes	SHA1 Hash

SHA224 (0x04)

Calibration: SHA224(0x010203)=0x3917AAAAA61D81DEB93EF1C27EC64
7F126FB932894B7CAA9DF286193

Name	Size	Description
Hash	28 Bytes	SHA224 Hash

SHA256 (0x05)

Calibration: SHA256(0x010203)=0x039058C6F2C0CB492C533B0A4D14EF7 7CC0F78ABCCCED5287D84A1A2011CFB81

Name	Size	Description
Hash	32 Bytes	SHA256 Hash

SHA512 (0x06)

Calibration: SHA512(0x010203)=0x27864CC5219A951A7A6E52B8C8DDDF 6981D098DA1658D96258C870B2C88DFBCB51841AEA172A28BAFA6A79731 165584677066045C959ED0F9929688D04DEFC29

Name	Size	Description
Hash	64 Bytes	SHA224 Hash

Public Keys (0x04)

ECDSA secp256k1 Uncompressed (0x01)

Name	Size	Description
Curve Point X	32 Bytes	Curve Point X on secp256k1
Curve Point Y	32 Bytes	Curve Point Y on secp256k1

ECDSA secp256k1 Compressed (0x02)

Point Y is 0x03 if Y is odd, and 0x02 if even.

Name	Size	Description
Curve Point X	1 Byte	Curve Point Y on secp256k1
Curve Point Y	32 Bytes	Curve Point X on secp256k1

Signatures 0x05

ECDSA secp256k1 (0x01)

Name	Type	Description
Total Size	Unsigned Byte	Total size of signature.
Signature	Array<Byte>	ECDSA Signature with the secp256k1 curve.

For the most up to date objects, please visit: `https://github.com/XYOracleNetwork/spec-coreobjectmodel-tex`.