Blockchain Interoperability Survey

1 Background

As crypto has gained more users, so too has the number of blockchains increased, numbering in the hundreds at the time of writing. While this increase has corresponded to a growing number of users, transactions, and features, it has also resulted in increased fragmentation of capital, users, and developers across blockchains and ecosystems.

The broad topic of blockchain interoperability is an ongoing area of research. While initial interoperability solutions focused on relatively narrow tasks like bridging assets, as usage increased over time they broadened in scope to providing cross-chain liquidity. At the time of writing, a growing number of protocols have launched with the aim of solving the broader problem of generalized message passing between chains.

This survey will provide a framework for classifying current interoperability protocols, with a particular focus on generalized message-passing protocols. More broadly, the purpose of this survey is to explore various protocols with which UXD may integrate as a part of a broader cross-chain strategy.

1.1 The Interoperability Problem

Blockchains are by design siloed entities, isolated ledgers that record some form of state (UTXO, accounts & balances, notes & owners, etc). That state is in turn updated according to a set of deterministic rules to establish consensus among network participants about the validity of state updates. However, these consensus mechanisms only pertain to the state of an *individual* blockchain's state/ledger; this is because different Layer 1s implement different forms of security and consensus (e.g. Proof of Work vs Proof of Stake) and store state differently. Akin to cross-border transactions between disparate nations, if users on one blockchain are to be able to influence the state of another, there needs to be a translator to facilitate activity.

It is always worth keeping in mind that interoperability can be considered as a subproblem of the "Oracle Problem", the concept that blockchains cannot pull in or publish data to any external system. Interoperability is a subproblem in the sense that, rather than solve for arbitrary data reading and writing (as is the goal of Chainlink), these protocols aim to solve only *blockchain to blockchain* data passing and verification. However, as with the Oracle Problem, all such solutions *by definition* require interaction with off-chain parties.

Broadly, given a source chain (message sender) and a destination chain (message receiver), a system for inter-chain communication must perform three basic functions:

1. Listen to the source chain for requests to send information to the destination chain
2. Confirm the validity of the message being sent and agree with other network participants on its contents
3. Relay said information to the destination chain in a form that can be read by the destination chain

The chosen implementation of these three functions gives rise to the specific characteristics of the cross-chain protocol, including generalizability, adaptability, trust assumptions, security, speed, capital efficiency, liveness backup, and actor permissioning, among others.

Tech Stack Layer	Definition
Base Layer	The infrastructure installed on source and destination chains that ultimately stores the result of read/write requests in the form of state and cross-chain messages
Authentication Layer	Protocol-level logic that verifies the validity of messages from other consensus environments
Transport Layer	Logic that defines how messages are routed, sent, and delivered between blockchains
Interface Layer	Message format and interpretation

We propose to use these factors, summarized in the table below, as the basis for evaluating and comparing cross-chain protocols.

Criterion	Definition
Generalizability	The ability of the cross-chain protocol to handle generalized message passing and contract calls
Adaptability	The ease with which a cross-chain protocol can be integrated with new blockchain types
Trust Assumption	Either (i) trust-minimized, meaning no additional trust assumptions beyond the connected chains are introduced; or (ii) trusted, requiring some level of trust on off-chain entities
Security	The incentives and guarantees of relevant actors in the system to ensure ultimate preservation of funds and/or correctness of messages passed
Speed	The speed with which the cross-chain protocol is able to process and finalize requests
Capital Efficiency	The amount of capital that must be locked up in order for the cross-chain protocol to maintain its other essential properties
Liveness Backup	The ability for the cross-chain protocol to provide alternative options in the case of simple liveness failures
Actor Permissioning	The permissioning requirements of the various actors in the cross-chain protocol; related to but distinct from Trust Assumption

At the end of the day, most of the above properties can be derived from logical implications of the structure of the "Cross-Chain Operators", i.e. the parties that actually "control" the cross-chain protocol. These commonly take the form of multisig committees, decentralized validator sets, decentralized oracle sets, or light clients among others.

In evaluating the technical stack of each cross-chain protocol, we borrow from Axelar and define the Base Layer, Authentication Layer, Transport Layer, and Interface Layer.

Note that not all cross-chain protocols need have all of the above layers. Some, such as LayerZero, are best understood as modules with the stack that are ambivalent towards the specific implementation of the rest. With these broad classifications and evaluation criteria in mind, the next section provides an overview of the current interoperability landscape.

1.2 Generalizability

Blockchain interoperability protocols vary in their specificity and flexibility; one may therefore consider a hierarchy of cross-chain protocols from least to most flexible. On the most specialized end

of the spectrum, asset-specific bridges transfer fully-collateralized wrapped assets from one chain to another. Such bridges are relatively simple to implement. However, they often sacrifice security when compared to more general protocols and but often require separate implementations on each connected chain. Generally these protocols bridge the most disparate blockchains to each other (such as BTC to ETH or ETH to SOL).

More general are application-specific protocols which aim to perform a specific functionality across a range of assets and chains. Members of this category include AnySwap and THORchain, which are designed for generalized inter-chain swaps.

Most flexible are generalized message passing protocols. These protocols aim to support arbitrary data transfers between connected chains. Recently, several previously application-specific protocols including Synapse and AnySwap have announced expansions in this area. Due to their structure as networks of nodes or validators with endpoints on connected chains, protocols of this type typically benefit from network effects as they scale. Within this category, protocols take different approaches to navigate tradeoffs between security, speed, flexibility, decentralization and scope. IBC, for example, is limited to chains with finality guarantees. Other protocols such as Axelar and LayerZero aim to support interoperability with a broader range of chains. These protocols will be explored in more detail using the framework in the previous section.

2 Security Mechanisms

Below we describe several common categories of such approaches. We note that these categories are simplifications. Within each category protocols exhibit significant variation in parameters and implementation, correspondingly large variation in security and technological properties.

Further, we note that even if the underlying messaging protocol is secure, smart contract bugs can still pose large risks. Indeed, several recent exploits such as the Nomad and Wormhole hacks occurred as a result of flaws in the smart contract endpoints rather than an attack on the underlying consensus mechanism. Moreover, the sheer economic activity that occurs in many bridges makes them a prime target for nefarious actors.

2.1 Light Clients with State Verification

This category consists of protocols which verify the state transitions of the source chain on the destination chain. Commonly, this is accomplished by providing a validity proof such as a ZK-proof alongside transaction data. Another approach employs fraud proof systems to dispute the validity of the state. Note that these bridges tend to be "trust-minimized" and are all currently implemented as optimistic rollups.

2.2 Clients with Consensus Verification

Protocols of this type do not attempt to prove or dispute the validity of the state directly[1]. Instead, these interoperability protocols accept messages as valid if consensus is reached on the source chain and the destination chain is able to verify this consensus. This may involve verifying the signatures of the most recent validator committee, or checking the state of the longest chain, for example.

One example of this category is Cosmos' IBC, which utilizes light clients to directly verify the signatures of other integrated Cosmos chains. Because protocols rely on validator consensus rather than fraud or validity proofs directly, they remain vulnerable to any attacks which compromise validator consensus on the source chain. Many members of this type are vulnerable to hard forks, although some consensus algorithms such as Tendermint, which provides instant finality, aim to limit this risk.

Members of these first two categories generally have strong security, inheriting guarantees from the source chain. Furthermore, protocols of this type tend to be capital-efficient (do not depend on *economic* security) and allow for flexibility in message passing. However, these types of approaches can also be difficult to scale. This is because they require implementation of new smart contracts on

[1]This is generally an incredibly complex problem as the chains need to directly understand each other's VMs or ZK proving systems.

the source and destination chains and specific implementations for each consensus pairing. The need to rewrite contracts depending on the consensus mechanism employed further limits the adaptability of these approaches.

2.3 Optimistic External Validation

In contrast to the previous approaches mentioned, members of this category employ external actors to ensure security. In a manner similar to the functioning of optimistic rollups, these protocols have some "watcher" actor that checks the validity of messages and submits a fraud proof within a certain challenge window period (usually around 30 minutes).

Although this gives a nice 1-of-n honest minority assumption (as only one watcher needs to submit a fraud proof), proponents of this model generally do not clarify that really one watcher needs to be *honest and online* during the challenge window . Given that these windows are relatively short, it remains to be seen how secure this model may be in practice. Of course, larger sets of watchers and longer challenge windows increase protocol security, though the latter at the expense of latency. For reference, Nomad, the current foremost optimistic messaging protocol employs a 30 minute challenge window.

2.4 External Validator Set with Consensus

This category is the broadest and contains the majority of current bridge and interoperability protocols. Protocols of this type rely on a set of external actors outside the source and destination chain to ensure security and verify transaction validity.

There are a variety of approaches within this category, from intermediate chains (Axelar) to sets of off-chain actors (Multichain, Synapse, LayerZero), to external validators from other chains which also validate messages (Wormhole). External actors establish consensus through methods including multisigs and threshold signature schemes.

Members of this category may also employ economic incentives to increase security. Two common forms of this are bonding and insurance. With bonding, validators must stake collateral of some kind, and forfeit this collateral upon proof of misbehavior. Generally this collateral is burned. This is rather odd, upon second thought, as this same collateral could be used to reimburse users who suffer losses as a result of this misbehavior. In the case of insurance, collateral is slashed to reimburse users in the case of lost funds.

It is quite difficult to understand the true economic benefit to attacking a cross-chain protocol, but a reasonable framework is to use "Max Attackable Value" or "MAV", which UXD Protocol defines as:

$$\text{MAV} = \max_{t} (\text{Value that can be stolen at time } t)$$

MAV is lower for protocols that only handle *flows*, i.e. cross-chain messages and swaps that do not require idle liquidity within the protocol, while MAV is higher for protocols that have large *stocks* of capital. In particular, the size of MAV influences the size of capital that must be held/bonded by validators in order to maintain economic security. We therefore take MAV/total bonded capital as a strong heuristic for the economic security of the chain.

We also note that in the case of general message-passing protocols without locked assets, there may still be economic incentives for attacks to occur depending on the content of the messages being sent and the counterparties/applications with which they interact on each chain. These can be difficult to quantify due to the complexity of the systems involved. Determining the necessary level of capital to be held to disincentivize bad behavior from off-chain actors is thus not straightforward.

The type of collateral required here also affects the security of the system; protocols using their own tokens as collateral are vulnerable to death spirals if token value plummets. Additionally, if the collateral asset differs from the asset being sent, further vulnerabilities may arise via attacks on price oracles linking the two. We also note that while these methods improve security, they decrease capital efficiency as ever-larger collateral values are required as message throughput increases.

Although members of this category vary in their security depending on the validation mechanism employed, they are often generalizable, easily scalable, and have low latency.

2.5 Locally Verified External Validator Set

Finally, we wish to note a special case of external validator sets, e.g. locally verified protocols. In these protocols, only the parties involved in a given transaction need to verify the transaction. This is in contrast to previous types, which generally require consensus from an entire external validator set.

This setup introduces further assumptions about the two parties involved[2]. Examples include Connext and Hop, and these protocols are commonly referred to as "Liquidity Networks", since they can be thought of as "over-the-counter" deals with parties such as market makers.

3 Current Protocols

Here, we describe and compare several prominent protocols for blockchain interoperability.

3.1 Abacus

Abacus is an inter-chain messaging protocol under development which relies on an external delegated PoS validator set. Abacus provides a simple on-chain API to send and receive inter-chain messages. The protocol relies on two simple smart contracts, "Outbox" and "Inbox", which then call relatively simple dispatch() and handle() functions. In particular, there is an outbox contract on every Abacus-supported chain, which stores a sparse Merkle tree of messages. There are $n - 1$ Inboxes on every Abacus-supported chain, one for each other chain supported. Messages are delivered via relayers to Inboxes by providing Merkle proofs against a signed Merkle root for the respective Outbox contract. The relayer processing function simple takes as inputs the signed Merkle root, the message, and a Merkle proof, and then sends the message to the recipient.

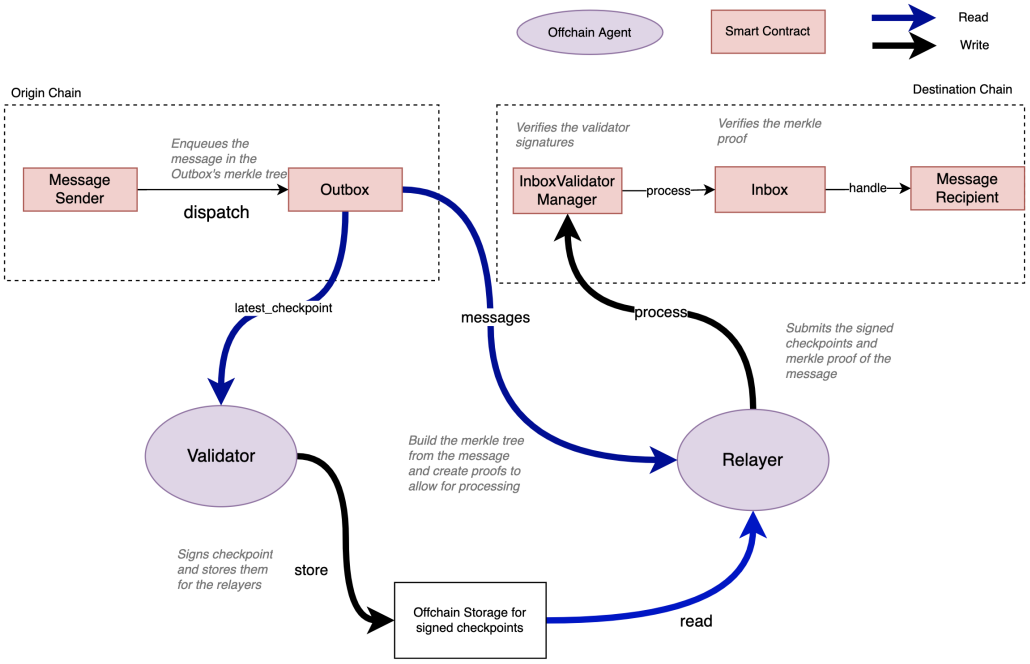

Figure 1: Overview of the Abacus messaging protocol

Regarding security, applications can also specify an external validator set of their own, with consensus from both validator sets required for transactions to be sent. This addition results in increased security should the native validator set be compromised, but also introduces additional liveness assumptions since the native validator set needs to sign off on transactions. Moreover, each Abacus-supported

[2]Namely, their economic incentives are either (a) adversarial and net-zero, meaning they cannot collude to take funds from the broader protocol as it would not make economic sense to do so or (b) isolated to their transaction, meaning there is no possible way to interact with the funds in the greater protocol.

chain requires its own validator set, which may fragment security, as well as a constant updating procedure to understand the structure of the validator sets on all other chains.

Validators are required to stake ABC tokens, and users may delegate said tokens, though both require a 21-day unbonding period. Stakers are rewarded with ABC token inflation, and are slashed if they attempt to censor messages (i.e. validator signs anything other than a valid Outbox Merkle root). Anyone may present evidence of a false message (referred to as "Watchtowers"). Anyone will be able to operate a validator, relayer, or watchtower.

In contrast to other protocols, rather than signing individual messages, validators sign checkpoints, which are just the Merkle roots corresponding to all messages from all applications of the connected chain.

3.2 Axelar

Axelar is a generalized message passing protocol which relies on a decentralized set of permissionless network validators using delegated PoS. Although the protocol claims this will support any-to-any communication, regardless of consensus mechanism or message payload, currently only EVM and IBC chains are supported. At the time of writing[3], Axelar is connected to 23 chains. There are currently 48 validators, with a total of $97.31M in TVL.

Validators run the cross-chain gateway protocol, which is a multi-party cryptography protocol (via threshold signature accounts on each L1) that sits on top of standard L1s, and are able to read/write messages and initiate contract calls. Essentially "gateway" contracts sit on every L1 and are monitored for incoming transactions which are read by the validators before being validated and written to the destination chain's gateway contract.

The on-chain state consists of cross-chain transactions and addresses on the source and destination chains. One limitation of the Axelar construct is that the destination contract address must implement the `IAxelarExecutable` interface in order to call the `_execute()` function. Arbitrary logic may be defined within this `_execute()` function.

Gateway-gateway processing takes about 120 seconds in the current implementation. Developers can solve for things like cross-chain atomicity by implementing additional features like nonced cross-chain execution and send-ack patterns to synchronize states across contracts. As shown in Figure 2, this ultimately allows for a simple hub-spoke model with central dApp logic on an optimized chain and satellite contracts elsewhere.

Validators choose which chains to support but must explicitly run node software on source and destination chains. Validators verify all cross-chain activity by observing the state of the source chain and voting to establish transaction confirmations between the source and destination chains and Axelar. Validators stake AXL tokens and receive inflationary rewards and transaction fees (though payments will not need to be explicitly in AXL). In the case of misbehavior (loss of liveness, double signing, etc.), validators' collateral can be slashed.

Axelar also relies on a set of off-chain relayers to perform less sensitive tasks such as coordinating validators to begin votes or monitoring on-chain addresses. Relayer architecture is flexible and permissionless.

[3]For current metrics, please refer to the dashboard here.

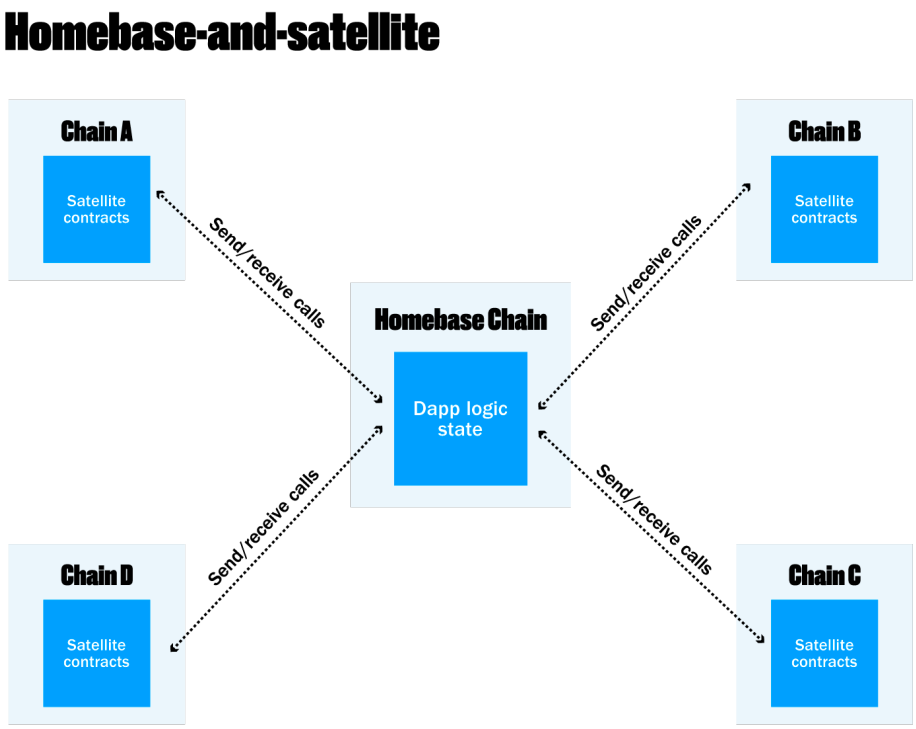

Figure 2: The Axelar Hub-Spoke Model

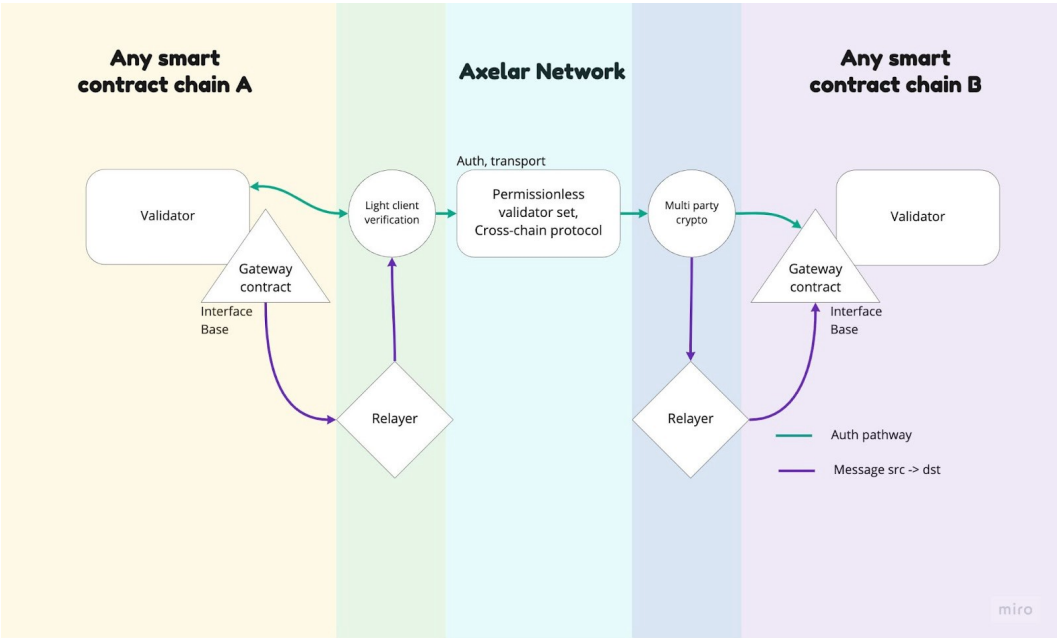

Figure 3: Overview of the Axelar messaging protocol

3.3 Chainlink CCIP

Chainlink has recently announced a decentralized protocol for generalized cross-chain messaging called CCIP. CCIP relies on an upcoming set of efficiency upgrades to Chainlink's off-chain reporting protocol (OCR). This set of upgrades, termed OCR 2.0, will allow for the network to run an increased number of nodes.

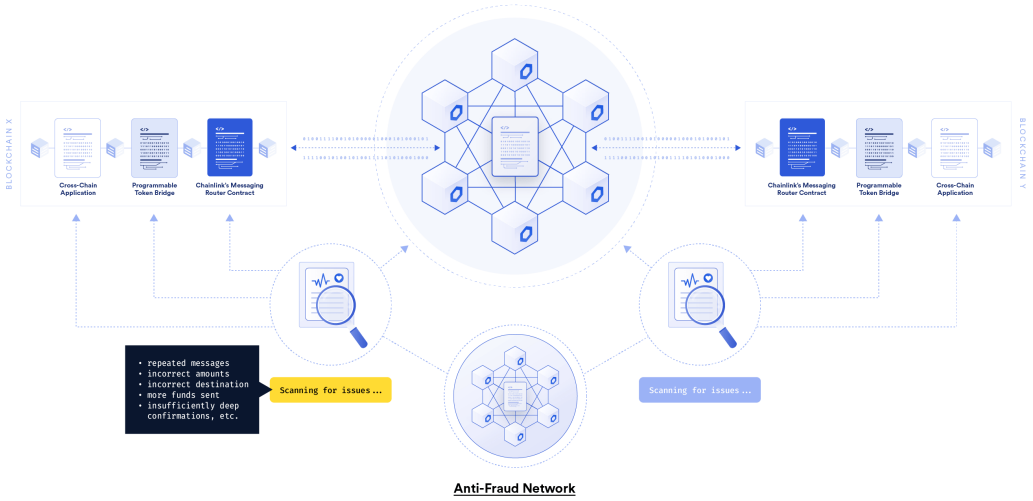

Figure 4: Overview of Chainlink's CCIP messaging protocol

Figure 4 contains an overview of CCIP's high-level structure. Under CCIP, messages are routed from smart contracts on source/destination chains to Chainlink's node network. From available documentation, messaging is to work similarly to the current Chainlink oracle transmission model. In this model, Chainlink employs oracle and external adapter contracts to read and transmit data to the node network. Nodes are permissioned, are required to stake LINK tokens, and receive shares of fees from relaying executed messages.

In addition to the OCR consensus mechanisms mentioned previously, CCIP plans to include a secondary security layer which it terms the "Anti-Fraud Network" (AFN). This AFN is essentially a set of decentralized oracle node committees which monitor the CCIP network for malicious activity and submit heartbeat messages when the system is operating normally. Detection of attacks or pauses in heartbeats result in an emergency shutdown of the relevant cross-chain services.

Although the AFN will initially consist of Chainlink nodes, the protocol will eventually allow for dApps satisfying certain staking criteria to operate nodes of their own. Because the protocol relies on oracles for consensus as opposed to proofs originating from the source chain, protocol security is dependent on the security of the underlying oracle and AFN. As such, the protocol remains vulnerable to attacks on either of these two sets or collusion between them.

3.4 IBC

The Inter-Blockchain Communication protocol (IBC) is an open-source interoperability protocol which provides generalized messaging between blockchains with instant finality. Notably, IBC is the protocol employed by the Cosmos ecosystem.

The IBC protocol consists of two primary layers. At the lowest level is the transport layer, which establishes secure inter-chain connections and authenticates data packets between blockchains. On top of this layer, the application layer specifies the interpretation and packing of data packets and is defined by protocols.

Figure 5 contains an overview of the message life cycle under IBC. Within the transport layer, authentication and verification is handled by light clients connected to validators on the consensus layers of connected chains. When a module on the source chain sends a data packet to the destination

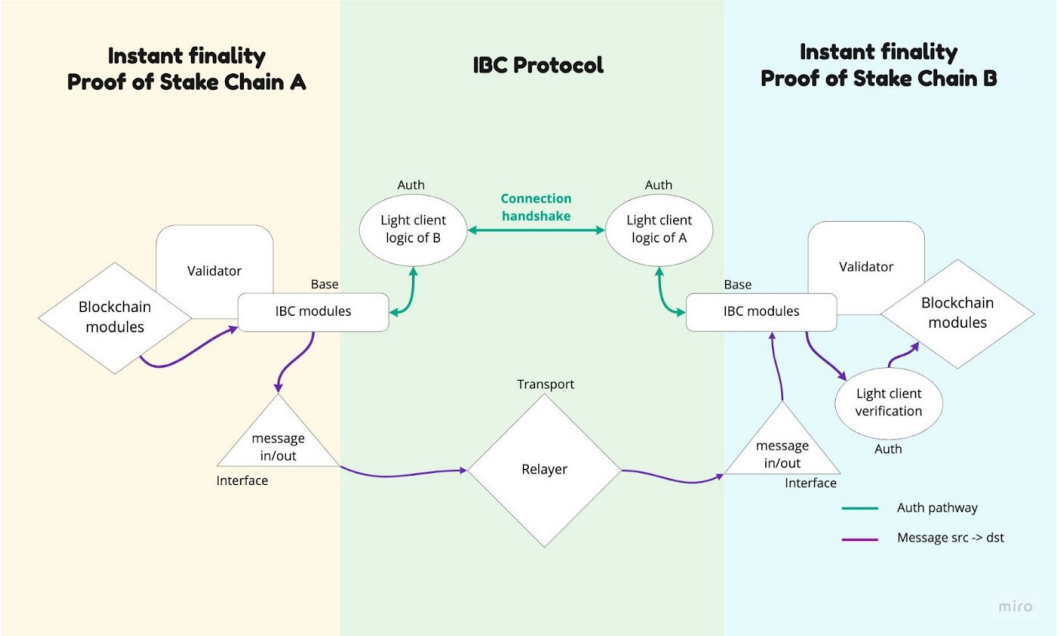

Figure 5: Overview of the IBC messaging protocol

chain, it first stores an on-chain commitment proof linked to the packet information. Light clients maintain a connection handshake between the connected chains and transmit proofs to the destination chain. Off-chain relayers, which have access to full nodes on both chains, transmit message data between the IBC modules on each chain. Once the destination chain verifies the source-chain proof, light clients store message receipt data and send an acknowledgement to the source chain. Finally, the destination chain executes logic based on the transmitted message.

Because IBC relies on light clients that verify consensus directly, it is among the most "trustless" of current interoperability protocols. These security guarantees come at the expense of flexibility; connected chains must run light clients at the base layer and have instant finality in order to be compatible. Because of instant finality, attacks compromising liveness do *not* compromise the security of IBC. However, if relayers fail or are attacked, IBC generally requires social coordination between relayers to reestablish liveness.

3.5 LayerZero

In contrast to previously mentioned protocols, LayerZero relies upon two types of off-chain entities called "Relayers" and "Oracles" for security. In effect, the two function similarly to a 2-of-2 multisig. While Relayers pass message information and transaction proofs between chains, Oracles pass block headers between source and destination chains, to guarantee the agreement between Oracles and Relayers. Oracles and Relayers are required to be independent for the described security guarantees, but specific implementations of either are left as an exercise to the user; theoretically, users and protocols could each provide their own implementations.

Both Relayers and Oracles receive a share of the transaction fees paid by the user on the origin chain. Currently Oracles are centralized and run by firms including FTX, Sequoia and Polygon. LayerZero has announced intent to use Chainlink as the default Oracle in the future.

For clarity, LayerZero is best understood as a specific implementation of a "Base Layer" from the Tech Stack description in Section 1. This is because LayerZero allows the user to specify how the Relayer/Oracle relationships may function (though default suggestions may of course be provided), as well as the particular form of messages to be sent. Stargate, the cross-chain token transfer protocol, is an application built using LayerZero as its lower-level messaging protocol.

The interface for LayerZero is a lightweight, on-chain client referred to as the LayerZero Endpoint. One Endpoint resides on each supported chain. Endpoints in turn consist of three modules: a

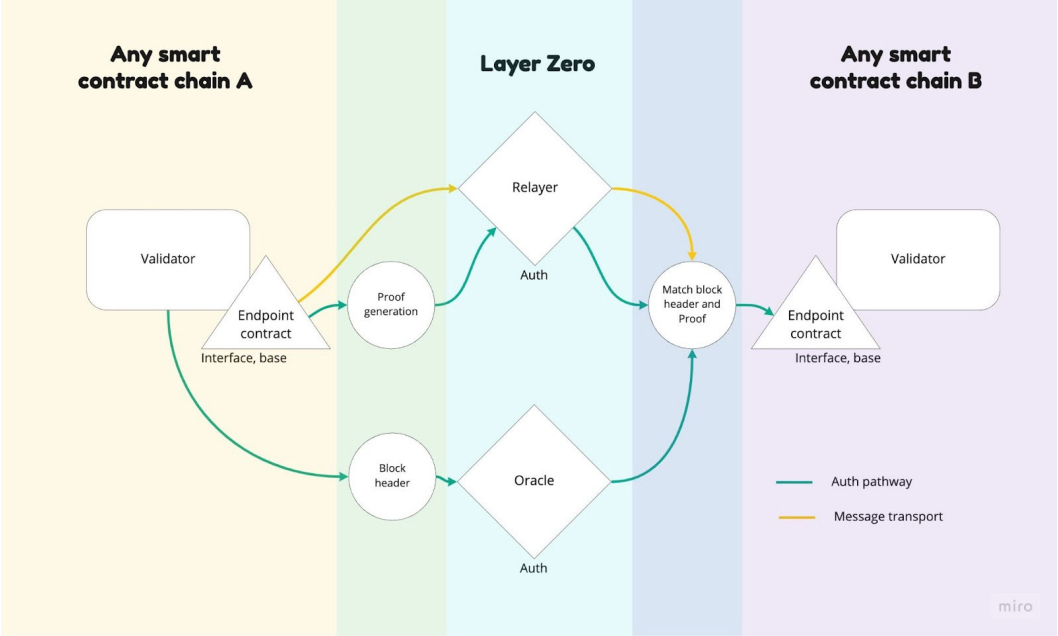

Figure 6: Overview of the LayerZero messaging protocol

communicator, validator, and network module. To allow for increased flexibility, endpoints can be extended via linked smart contracts containing chain-specific communication protocols called "Libraries". Although Libraries are immutable once implemented, the protocol allows for flexibility by permitting applications to add new Libraries and specify which Library to use.

While LayerZero supports generalized message passing, as noted above it does not specify the Interface (form of message) that must be passed. Applications interact with LayerZero by implementing methods corresponding to send and receive functionality.

Though the LayerZero whitepaper was initially published in May 2021, the ecosystem is still developing. At the time of writing, LayerZero connects to 7 EVM chains, with stated plans to expand to others including Cosmos and Solana. On the application layer, the LayerZero dAPP ecosystem currently consists of the Stargate app, a cross-chain routing and liquidity protocol with $468M in TVL. We were unable to find further information on the current number and operators of Oracle and Relayer nodes, since notably these seem to not be terribly public.

In general, LayerZero's biggest critique is around the implementation of the Oracle and Relayer parties, as allowing for "general" implementation is in some sense saying that a default implementation is not obvious. Further, it is not clear how to prevent forms of collusion between these parties.

3.6 Multichain

Formerly known as AnySwap, Multichain launched in July 2020 as an asset bridge and router. Multichain relies on an off-chain MPC network for security, each node of which independently verifies the status of the original chain and uses a threshold consensus algorithm. It does not appear that this validator set is permissionless. Multichain offers traditional lock-burn bridging, with the associated smart contract and validator risks, as well as a liquidity router using anyTokens as intermediary tokens (such as anyUSDC), and a cross-chain contract calling service, anyCall.

The message-passing protocol underlying anySwap, anyCall, allows for cross-chain message passing. Messages are sent and received by endpoints on connected chains, which transmit via the MPC network. As noted above, Multichain uses an MPC TSS scheme of validator nodes that watch the state of the source chain. Similar to Axelar, the anyCall endpoints contain an `anyExec()` function that implements the instructions sent through the anyCall protocol. The workflow is as follows: someone initiates a call to a sender contract on the source chain $\implies$ `anyCall(source chain)` is called $\implies$ MPC Network sees `anyCall(source chain)` $\implies$ `anyExec(destination chain)`

$\implies$ `anyCall(executor)` $\implies$ `anyExec(destination chain)`. The protocol allows for gas fees to be paid on either the source or destination chains.

For example, the first integration of anyCall was with Curve, in which Curve's gauge mechanism is generalized by allowing cross-chain CRV rewards deployment. For example, it allows users to claim wrapped CRV on Fantom via a rewards redemption request that is processed on Ethereum.

Intuitively the (anticipated) functionality of anyCall and Axelar seem quite similar, with the key differences being (i) Multichain not requiring staking (more capital efficient, but relies on reputation of MPC nodes) and (ii) Axelar having a more diverse, permissionless validator set (less capital-efficient and requires economic security).

The Multichain SMPC network currently[4] consists of 23 nodes connected to 63 chains, both EVM and non-EVM, and is used by dApps including Curve and Hundred Finance. At the time of writing, the network has $2.51B of TVL, with a daily average volume of $54M in assets.

3.7 Nomad

Unlike the aforementioned protocols, Nomad operates an optimistic model instead of a consensus model. This functions similarly to how optimistic rollups operate today. The Nomad protocol consists of off-chain relayers and on-chain endpoint contracts for sending and queuing messages.

Figure 7 contains a high-level summary of the Nomad messaging protocol's structure. Messages are sent from the origin chain via a Home contract, where they are put into a queue. Similar to Abacus, every chain must implement a Home contract, and the ultimate source of truth with messages to be sent is stored in a Merkle tree. Replica contracts for $n-1$ connected chains, correspond to the Home contracts on the other $n-1$ connected chains. and hold knowledge of the current state root and the identity of the Updater. Each Home contract acts as a broadcast channel, emitting a general broadcasted event, meaning that all k Replica contracts on other chains referencing said Home contract can receive updates at once[5]. Currently, the Updater can only be set via a function that belongs to Nomad governance, but we suspect that this will be done in a more trustless way in the future.

On the security model, messages signed by updaters can be rejected by any Watcher that submits a fraud proof during a 30-minute window. This setup gives a nice 1-of-n honest minority assumption, as only one watcher needs to submit a fraud proof. However, proponents of this model generally do not clarify that really one watcher needs to be ***honest and online*** during the challenge window period. Given that these windows are relatively short, it remains to be seen how secure this model may be in practice.

We also note that the calculation of the optimistic window depends on the blockchains involved and theoretically may need to be more individually tailored for different (Home, Replica) pairs due to differences in block and confirmation times. If a malicious actor can block the submission of a fraud proof to the Home chain for longer than this window, their message may execute on the Replica chain.

Currently, watchers are permissioned and centralized since the current system lacks mechanisms to prevent false challenges. Fraud occurs when an Updater signs a malicious Merkle root update with an inserted message. This fraud can occur in two distinct ways. First, the Updater may commits two new roots with the same previous root, known as double signing. Second, the Updater may commit an invalid update.

Although the former is easy to detect, the latter requires referencing the queue of messages in the Home contract and noticing that the root is not contained in the proposed and calculated queue of roots. However, while this sort of fraud can be proven in the Home contract (the source of truth for message origination), it cannot be directly proven for the Replica contracts to be halted. Nomad therefore relies on a permissioned set of Watchers on the Replica chain to pause message receipt. Once fraud is detected, the system must be restarted from a failed state. For economic security, the slashing of an Updater's stake must thus be sufficient to protect against ongoing griefing.

[4]For the curent status of the Multichain SMPC network, please refer to the dashboard here.

[5]There is potential for this broadcasting to be used for more general one-to-many blockchain communication such as notification systems.

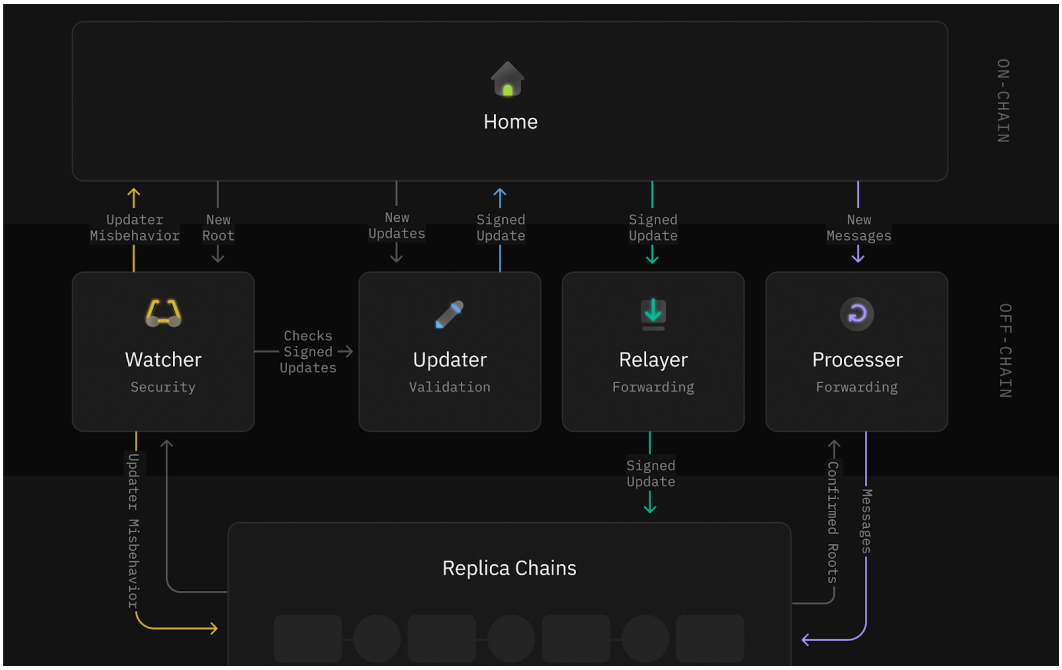

Figure 7: Overview of the Nomad messaging protocol

In August 2022 there Nomad suffered a hack of $190M due to a smart contract bug which allowed receiving contracts to bypass the message verification mechanism.

3.8 Synapse

Synapse is a cross-chain AMM and bridge protocol. While initially launched with a focus on asset bridging, in July 2022 the protocol announced a shift towards general message passing. As part of this transition, Synapse plans to launch Synapse Chain, an optimistic rollup settling on ETH with delegated PoS. However, the information on implementation specifics is still sparse. Synapse is unique in the sense that the SYN token is meant to also run the standalone blockchain, which is meant to accrue economic value in itself as an optimistic rollup. The specific rollup implementation is unclear, though from initial materials it seems likely that design choices will be similar to Arbitrum and Optimism, along with an initially centralized sequencer.

To transmit and verify transactions between chains, Synapse plans to rely on four on-chain actors:

1. Notaries - sign Merkle roots on connected chains and bond SYN to attestations
2. Broadcasters - forward updates between contracts
3. Guards - observe cross-chain messages and submit fraud proofs
4. Executors - post final transactions after the latency window

Note this setup functions fundamentally similarly to Nomad's implementation. First, a user sends a message on the source chain to a router contract and a centralized permissioned Notary signs an attestation of the updated Merkle root. Following this, the message is passed across to the Router contract on the destination chain, although it is unclear if this is one-to-many broadcasting. Guards check for fraud, and if none is detected the message is executed by an Executor on the destination chain.

Although some of the more nuanced points of optimistic verification for cross-chain messaging remain to be seen, it is clear that this security model is a significant improvement over the current Synapse multisig model.

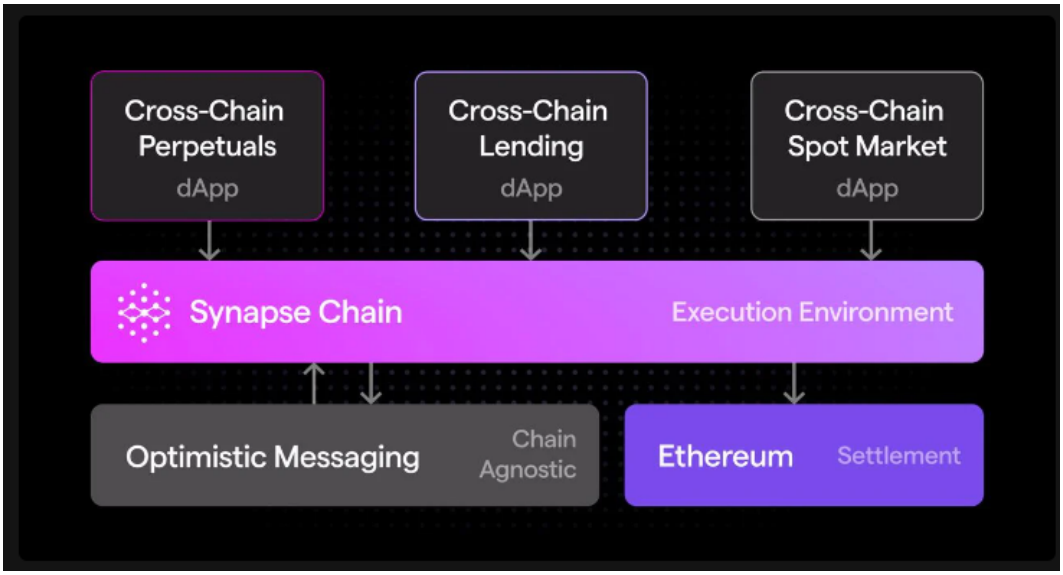

Figure 8: Overview of the anticipated Synapse tech stack

The Synapse bridge application is currently[6] connected to 16 chains and has a TVL of $206M with a daily volume of $8.41M. At the time of writing, the broader messaging protocol and linked chain has yet to be released.

3.9 Wormhole

Wormhole is a generic messaging protocol relying on an external validation set for security. The Wormhole network consists of 19 oracles, termed "Guardians", which correspond to permissioned PoS validators running full nodes of each connected chain. Developers deploy "xDapp" contracts on their source chain, which receive transactions from the end user. These contracts then interact with "Core" contracts that contain the source-chain-deployed version of bridging and generalized message passing. However, the Relay contracts which handle generalized message passing are still in development. Currently, the Wormhole ecosystem contains an NFT bridge and the Portal token bridge which at the time of writing contained $493.12 TVL. It includes support for Solana, Terra, Ethereum, and Binance Smart Chain.

Figure 9 contains an overview of Wormhole's structure. Guardians are responsible for observing endpoints on all connected bridges and verifying transactions. Guardians are given equal weighting, are chosen in a centralized manner by the protocol, and are not required to post bond. Transactions are confirmed via a multi-signature scheme with a 2/3 threshold at the time of writing.

Again, Wormhole today currently operates at a 13-of-19 multisig on transaction verification and relies on the reputation of the parties running the guardians, such as Figment, FTX, and Staked. This trade-off has its benefits, however, as Wormhole is able to quickly integrate with different consensus-type chains because the fundamental tech to be used is quite simple (t-Schnorr signatures). Relayers do not need to be factored into the security model; instead they must just be capable of uploading messages to the blockchain.

In addition to Guardians, Wormhole will rely on a series of untrusted, publicly-run relayers to transmit messages from Guardians to the contracts on the destination chain. At the time of writing, these "Generic Relayers" are still under development.

In February 2022, Wormhole suffered a $325M hack due to a smart-contract bug in the Solana endpoint contract.

[6]For the latest statistics on the Synapse bridge network please refer to the Synapse dashboard here.

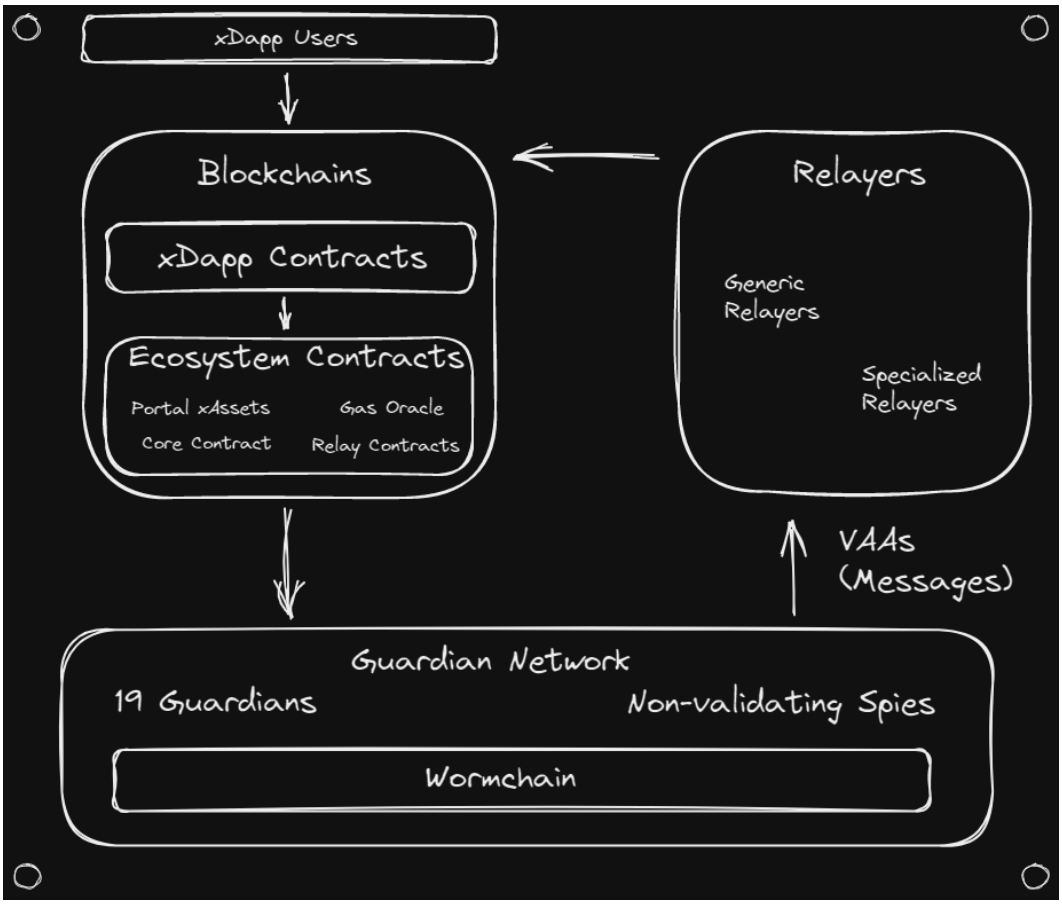

Figure 9: Overview of the Wormhole messaging protocol

4 Conclusion

Based on the above, UXD has generally adopted the framework that most of the cross-chain messaging protocols are fundamentally similar on a Base Layer level, in that they all generally function as deploy contracts on chain X and Y, contracts accept messages, someone validates/invalidates and relay messages to sister contract on chain Y, which then executes some instruction. We see the most meaningful differences at the authentication layer, and think it is still an open question about how to optimize for efficiency and security. For example, LayerZero can be thought of as a Base Layer and Transport Layer that is Authentication Layer agnostic, in the sense that any parties or sets of parties can be plugged into the Oracle and Relayer roles. For example, a decentralized network of PoS validators could function themselves as a Relayer in the LayerZero network. Other protocols like Axelar and Wormhole choose dPoS and Multi-Sig Authentication Layers, respectively. One thing to note that is not addressed much in any of the documentation of the respective cross-chain protocols is latency. Note that an absolute minimum amount of latency exists because the theoretical minimum is 1 block confirmation on chain X => message pass => 1 block confirmation on chain Y. Therefore, minimum latency is required to be someone on the order of several seconds to several minutes on most chains. This is important when evaluating something like the Optimistic solution offered by Nomad and Synapse, where there's a minimum latency of 30 minutes. However, given the nature of the protocol in a cross-chain world likely would not require frequent interaction (primarily mint/redeem or borrow/lend), a 30 minute latency in the Optimistic model may be acceptable. We will continue to explore these options as we develop our cross-chain strategy.

For UXD's purposes, we believe that either a LayerZero model in which we are able to set our own parameters (oracle/relayer, min block confirms, etc) or a more "out-of-the-box" solution like Axelar or Nomad may be suitable.

5 Disclaimer and Risks

All decentralized stablecoins carry risks related to their usage and price stability. Please review UXD Protocol's Risks section in the docs for more information on potential risks. `https://docs.uxd.fi/uxdprotocol/overview/risks`

The views expressed herein are for informational purposes only and, unless otherwise stated, reflect only the subjective views of the applicable speaker, which are subject to change. Nothing herein constitutes investment, legal, or tax advice or recommendations. This material does not constitute or form part of an offer to issue or sell, or of a solicitation of an offer to subscribe or buy, any securities or other financial instruments, nor does it constitute a financial promotion, investment advice or an inducement or incitement to participate in any produce, offering or investment. This material should not be relied upon as a basis for making an investment decision. It should not be assumed that any investment in the asset class described herein, or any company or asset described herein, will be profitable and there can be no assurance that future events and market factors would lead to results similar to the results discussed in this article. Any projections, estimates, forecasts, targets, prospects and/or opinions expressed in these materials are subject to change without notice and may differ or be contrary to opinions expressed by others. No representation or warranty, express or implied, is made as to the accuracy or completeness of the information contained herein. Finally, UXD Protocol has not formed any opinions on the above based on any relationships with the relevant protocols and conclusions therein rely only on our subjective opinions.