# Multichain Trustless Bridge - Draft WIP

John Whitton

October 2022

# Contents

# 1 Introduction

As of October 4th, 2022, crypto assets totalled $998 Billion with over 100 disparate chains holding assets.

A unifying vision of the distributed ledger technology is to connect multiple distributed systems (chains) together. However as these systems have been developed over $2 Billion has been stolen stolen across 13 separate cross-chain bridge hacks.[1].

Here we give an overview of various existing approaches to the bridging of assets between chains and expand on the Horizon Bridge design [2] to cater for bridging to etherum 2.0 [1] Proof of Stake. Revisiting light client construction, frequency of block Header synchronization, economics and incentives for relayers and multi-chain support.

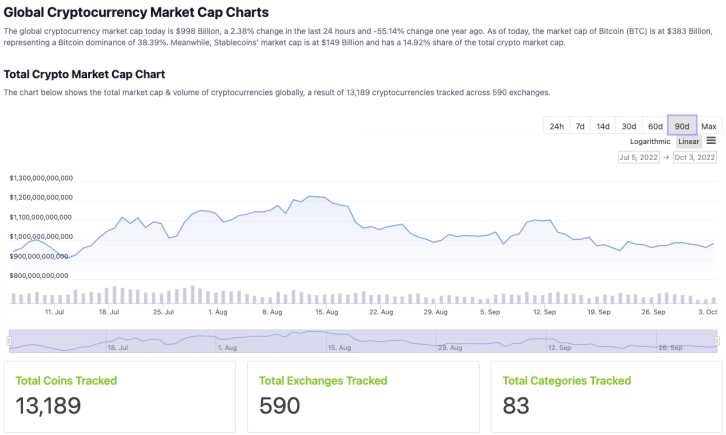


Figure 1: Crypto Market Capitalization

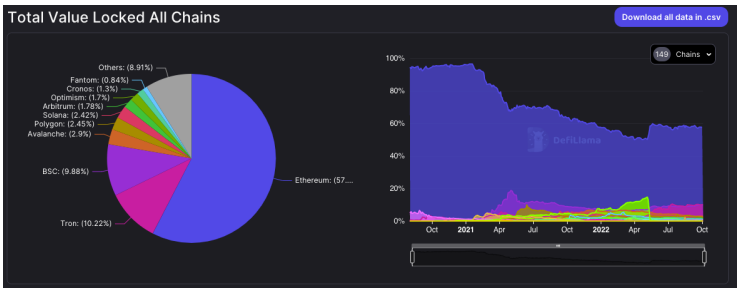


Figure 2: Total Value Locked by Chain

## 2 Our Contributions

We propose a top down design staring with the following components

1. **Smart Wallet:** A Multichain secure, frictionless, programmable non-custodial wallet infrastructure. [20]

2. **Universal Token Management:** provides the ability to track deposits via a token management layer which can be used to reduce gas fees and also allow idle funds to be leveraged. (similar to Ocean Protocol[23], Sushi's bentobox [22] and balancer's vault[21]).

3. **Dedicated Chain:** We propose a purpose driven relay chain which includes consensus support for destination chains similar to map's atlas chain [16] and additional on-chain infrastructure similar to wormchain [10].

4. **Multichain Bridging:** A hub and spoke model building with Guardians to validate the authenticity of the transactions and Relayers to relay the messages to the various chains (similar to wormhole [9].

5. **On Chain Components:** Smart Contracts to bridge tokens between chains and a Gas Management Layer to reduce gas fees and user friction by enabling users to pay gas across the multichain transactions using a single token. (similar to acala's flexible fees [24]).

6. **Zero Knowledge Proof Improvements:** We propose improving the cryptographic proof construction and checking of proofs as part of the light client process similar to map protocol [4]. And an ultralight blockchain client (similar to Plumo [3]).

# 3 Background and Related Work

## 3.1 Bridging Solutions

### 3.1.1 Horizon Bridge

Horizon Bridge is a gas-efficient, cross-chain bridge protocol to transfer assets from a BFT blockchain to another blockchain (e.g., Ethereum) which supports basic smart contract execution.[2]

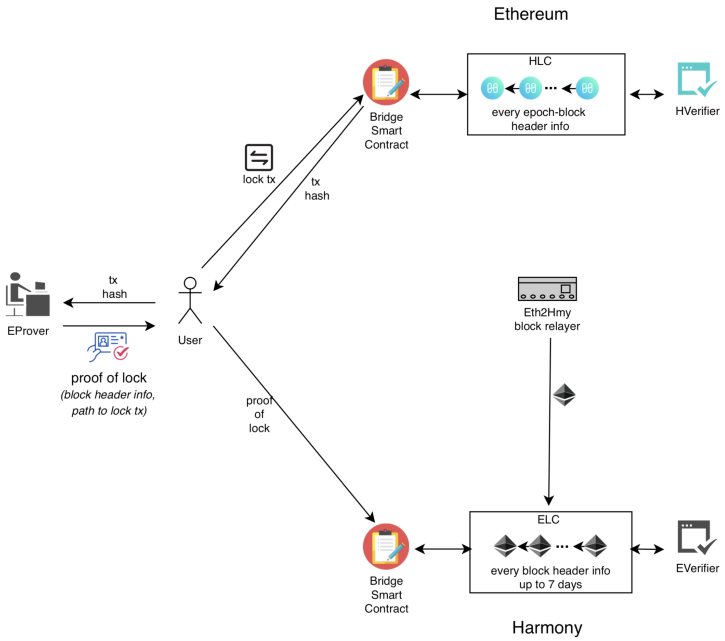


Figure 3: Horizon Ethereum to Harmony Flow

**Hack:** The initial version of the horizon bridge (not the trustless version currently under development [13]. Was hacked for $100 million in tokens when the private keys of the multisig holding the locked funds were compromised.[5]

### 3.1.2 Nomad

Nomad:is an optimistic interoperability protocol that enables secure cross-chain communication. [5] Optimistic verification doesn't use light clients or natively verify cross-chain messages. Instead, messages are optimistically signed on the origin chain, and a timeout period is enforced on the destination, during which the message can be inspected and vetoed if anything awry is noticed.

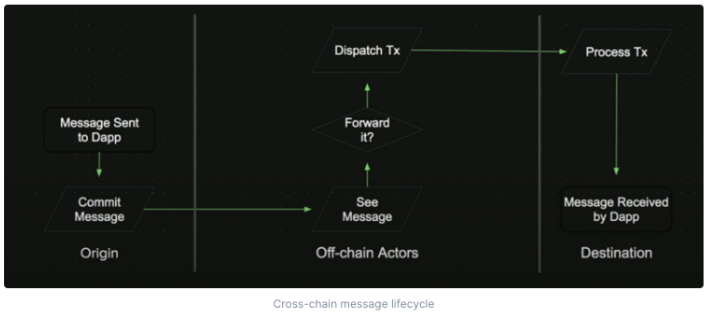


Figure 4: Nomad Cross Chain Message Life Cycle

**Hack:** Nomad was hacked for $190 Million after a hacker found a vulnerability in Nomad's code[14] that failed to properly validate that a message was approved before processing it.[6]

### 3.1.3 Map Protocol

Map Protocol is an Omnichain Layer of Web3 with Provably Secure Cross-chain Communication Built upon Light-client and zk-SNARK technology.[4] It includes it's own relay-chain [16] which embeds each destination chains consensus algorithm as precompiled contracts to facilitate light client construction.

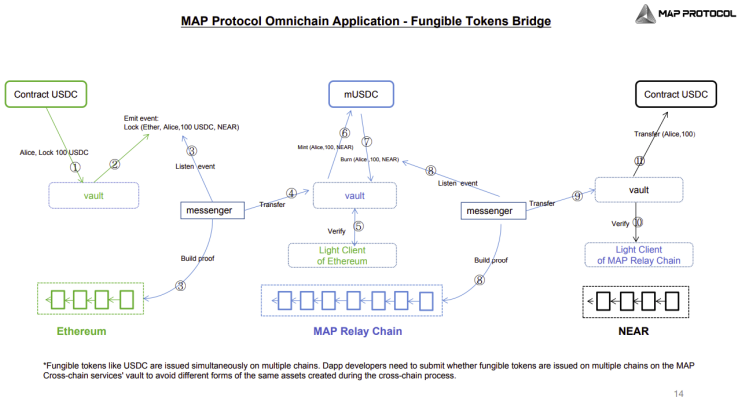


Figure 5: Map Protocol Fungible Token Flow.

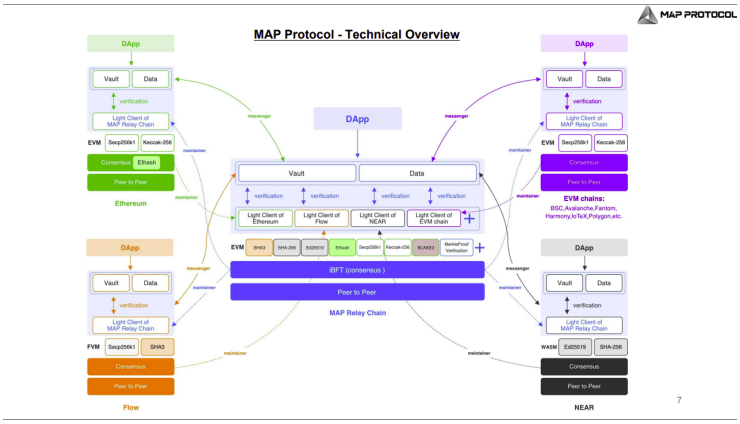


Figure 6: Map Protocol Architecture

**Hack:** As of Oct 4th, 2022 no hack of map protocol has been reported.

### 3.1.4 Wormhole

Wormhole is a generic message passing protocol that connects to multiple chains including Ethereum, Solana, Terra, Binance Smart Chain, Polygon, Avalanche, Oasis, Fantom, Karura, Celo, Acala, Aurora and Klaytn.[9].

Wormhole has a Core Bridge contract deployed on all the connected networks. Wormhole Guardians run a full node for each of the connected chains, specifically listening to any events from the Core Contracts. The core contracts emit a message, which is picked up by the Guardians. The Guardians verify the message and sign it, creating a VAA (Verified Action Approval). This VAA then sits on the Guardians network where it can be retrieved by the user or by a relayer to be submitted to the target chain to process the message. Unlike other bridges, a relayer in Wormhole has no special privileges, it's just a piece of software that shuttles messages between the Guardian network to the target chain, and is not a trusted entity.

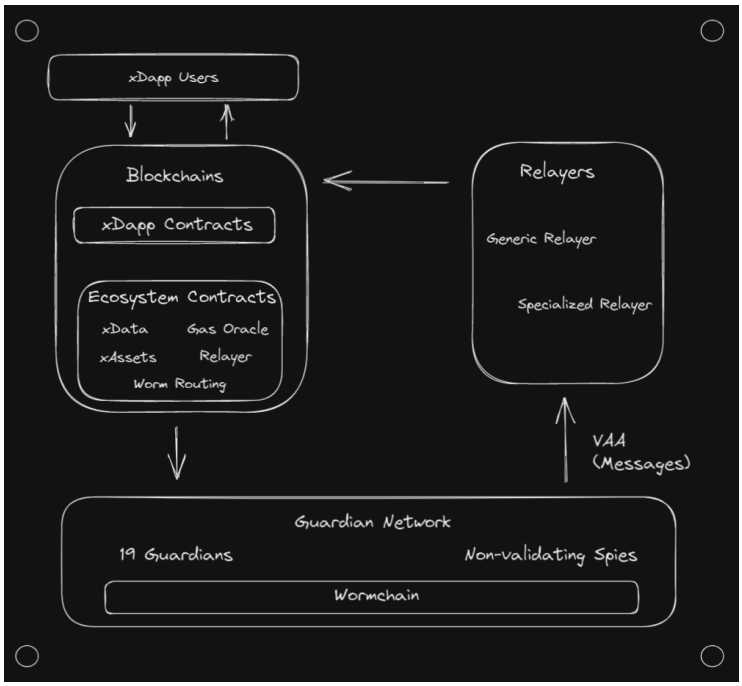


Figure 7: Wormhole Architecture

**Hack:** Wormhole[17]was hacked for $326 million after an attacker exploited the use of a deprecated, insecure function to bypass signature verification.[3]

## 3.2 Foundational Layers

### 3.2.1 On Chain Components:

- **XCLAIM**
- **Horizon**
- **Wormhole**

### 3.2.2 Light Clients:

- **Proof of Work:**
- **Proof of Stake:**
- **Bitcoin:**
- **Solana:**
- **Polkadot:**
- **Near:**

### 3.2.3 Bridging Components:

- Guardians :
- Relayers:

### 3.2.4 Communication Protocols

- **Cosmos IBC**
- **Polkadot XCMP:**

### 3.2.5 Zero Knowledge Proofs:

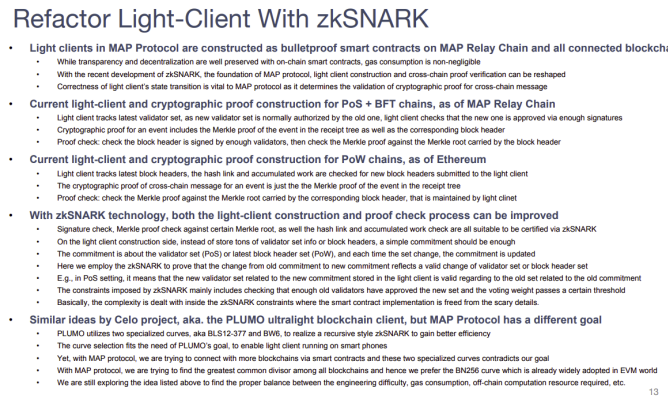


Figure 8: Map Protocol zkSNARK Improvements

**Main circuit**

We first define the following helper methods:

– $\mathsf{EncodeEpochToBits}(i, r, \delta, \delta', t, apk, \{pk_i\}_{i=1}^n)$ :
1. Encode $i$, the epoch index, as a 16-bit integer.
2. Encode $r$, the consensus round number, as an 8-bit integer.
3. Encode $t$, the maximum number of non-signers, as a 32-bit integer.
4. Encode $\delta$, the current epoch entropy, in 128 bits.
5. Encode $\delta'$, the parent epoch entropy, in 128 bits.
6. Encode each public key in $\{pk_i\}_{i=1}^n$ as a $\mathbb{G}_2$ compressed point. If there are fewer public keys than the maximum defined in the system parameters, pad with $G_2$ until the maximum number of public keys is reached.

– $\mathsf{EncodeEpochToBitsEdges}(i, \delta, \delta', t, apk, \{pk_i\}_{i=1}^n)$ :
1. Encode $i$, the epoch index, as a 16-bit integer.
2. If this is the first epoch, encode $\delta'$, the parent epoch entropy in 128 bits. If this is the last epoch, encode $\delta$, the current epoch entropy in 128 bits.
3. Encode $t$, the required signer threshold, as a 32-bit integer.
4. If this is the last epoch, encode $apk$, the aggregated public key of this validator set, as a compressed $\mathbb{G}_2$ point.
5. Encode each public key in $\{pk_i\}_{i=1}^n$ as a $\mathbb{G}_2$ compressed point. If there are fewer public keys than the maximum defined in the system parameters, pad with $G_2$ until the maximum number of public keys is reached.

Next we describe the main circuit. In the following let

$$E_j = \{i_j, r_j, \delta_j, \delta'_j, t_j, apk_j, \{pk_{j,k}\}_{k=1}^n\}$$

A subroutine taking as input some $E_j$ is assumed to discard those elements included in it which are not a part of the subroutine's input.

– $\mathsf{MainCircuit}(H'(e_1), H'(e_N) : \sigma_{agg}, \{H(e_j)\}_{j=2}^N, \{\mathbf{b}_j\}_{j=1}^{N-1}, \{E_j\}_{j=1}^N)$:
1. For each $j = 2...N$ perform:
   (a) Check that $apk_{j-1} =? \sum_{i=1}^n b_i \cdot pk_{j-1,i}$ where $b_i$ is the $i$-th bit of $\mathbf{b}_{j-1}$.
   (b) Check that $\delta_{j-1} =? \delta'_j$
   (c) Check that $i_{j-1} =? i_j + 1$
   (d) Encode $E_j$ as $e_j$ using $\mathsf{EncodeEpochToBits}$ and hash it using $\mathsf{BHPedersenHash}$. Then, run $\mathsf{Blake2Xs}$ on the intermediate result to obtain the final result of the composite hash. Finally, complete the hash following the hash-to-group method described in Sections 5 and 6.1. Check that the result is equal to $H(e_j)$.
2. Check that $apk_N =? \sum_{i=1}^n pk_{N,i}$.
3. Check that $e(\sigma_{agg}, G_2^{-1}) \cdot e(H(e_2), apk_1) \cdot ... \cdot e(H(e_n), apk_{n-1}) =? 1_{\mathbb{G}_T}$
4. Encode $E_1$ as $e_1$ and $E_N$ as $e_N$ each using $\mathsf{EncodeEpochToBitsEdges}$. Hash individually both $e_1$ and $e_N$ directly with $\mathsf{Blake2s}$. Tightly pack, individually, the first and last epoch resulting hash bits into elements of $\mathbb{F}$. Check that the results of this packing are equal to $H'(e_1), H'(e_N)$ respectively.

Figure 9: Plumo Main Circuit

### 3.2.6 Dedicated Chain:

# 4 Our Solution

Here we give an overview of bridging an ERC20 token from Ethereum to Solana, Polkadot and Harmony.

## 4.1 Smart Wallet

## 4.2 Universal Token Management

## 4.3 Dedicated Chain

## 4.4 Multichain Bridging

## 4.5 On Chain Components

## 4.6 Zero Knowledge Proof Improvements

- **Metadata Transaction:**
- **Metadata Relay:**

# References


[1] Maksym Zavershynskyi, 2020 *ETH-NEAR Rainbow Bridge* https://near.org/blog/eth-near-rainbow-bridge/

[2] Riongjan Lan, Ganesha Upadhyaya, Stephen Tse, Madhi Zamani (Harmony, Visa Research) 2020 *Horizon: A Gas-Efficient Trustless Bridge for Cross-Chain Transactions* https://github.com/harmony-one/horizon/blob/main/docs/assets/horizon-whitepaper.pdf

[3] Psi Vesely, Kobi Gurkan, Michael Straka, Ariel Gabizon, Philipp Jovanovic, Georgios Konstantopoulos, Asa Oines, Marek Olszewski, and Eran Tromer. 2021 *Plumo: An Ultralight Blockchain Client* https://eprint.iacr.org/2021/1361.pdf

[4] Map Protocol, 2022 *Omnichain Layer of Web3 with Provaby Secure Cross-Chain Communication* https://files.maplabs.io/pdf/mapprotocol_Litebook_en.pdf

[5] Nomad.xyz, 2022 *The Nomad Protocol* https://docs.nomad.xyz/the-nomad-protocol/overview

[6] ParityTech, 2022 *BEEFY (Bridge Efficiency Enabling Finality Yielder)* https://github.com/paritytech/grandpa-bridge-gadget/blob/master/docs/walkthrough.md

[7] Snowfork, 2020. *A trustless, general-purpose Polkadot ¡¿ Ethereum bridge.* https://snowbridge-docs.snowfork.com/concepts/

[8] Christopher Goes, 2020 *The Interblockchain Communication Protocol: An Overview* https://arxiv.org/pdf/2006.15918.pdf

[9] Wormhole Foundation, 2022 *Wormhole: a generic multichain message passing protocol* https://docs.wormhole.com/wormhole/

[10] Wormhole Foundation, 2022 *WormChain* https://book.wormhole.com/wormhole/8_wormchain.html

[11] Polkadot Wiki, 2020. *Cross-chain Message Passing (XCMP).* https://wiki.polkadot.network/docs/en/learn-crosschain

[12] Research at W3F, 2020. *XCMP Overview.* https://research.web3.foundation/en/latest/polkadot/XCMP/index.html

[13] Research at W3F, 2020 *HRMP Channels.* https://research.web3.foundation/en/latest/polkadot/XCMP/HRMP%20channels.html

[14] Polkadot Wiki, 2020. *Bridges* https://wiki.polkadot.network/docs/en/learn-bridges

[15] Parity Technologies, 2020. *Parity Bridges Common.* https://github.com/paritytech/parity-bridges-common

[16] Ruitao Su, 2020. *Bringing BTC to Polkadot: Acala x Ren.* https://medium.com/acalanetwork/bringing-btc-to-polkadot-acala-x-ren-e7959855d5aa

[17] ChorusOne, 2020. *wormhole: A Cosmos-Substrate bridge.* https://github.com/ChorusOne/wormhole-bridge)

[18] Ethan Buchman, University of Guelph, 2016 *Tendermint: Byzantine fault tolerance in the age of blockchains* https://atrium.lib.uoguelph.ca/xmlui/bitstream/handle/10214/9769/Buchman_Ethan_201606_MAsc.pdf

[19] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt, 2019 *XCLAIM: Trustless, Interoperable, Cryptocurrency-Backed Assets* https://eprint.iacr.org/2018/643.pdf

[20] Modulo.so, 2022 *modulo.so secure, frictionless, programmable non-custodial wallet infrastructure* https://modulo.so/assets/products.pdf

[21] Balancer, 2021. *Balancer The Vault* https://docs.balancer.fi/products/the-vault

[22] Sushi, 2021 *BentoBox* https://docs.sushi.com/docs/Developers/Bentobox/

[23] Shell Protocol, 2022 *The Ocean, Shell Protocol v2 White Paper, Part 2* https://shellprotocol.io/static/Ocean_-_Shell_v2_Part_2.pdf

[24] Acala Foundation, 2022 *Acala Flexible Fees* https://wiki.acala.network/learn/flexible-fees

## Documentation and Github References


[1] Vitalik Buterin, 2020 *Vitalik's Annotated Ethereum 2.0 Spec* https://notes.ethereum.org/@vbuterin/SkeyEI3xv

[2] Ethereum Github, 2022 *Consensus: Phase 0 – The Beacon Chain* https://github.com/ethereum/consensus-specs/blob/master/specs/phase0/beacon-chain.md

[3] Vitalik Buterin, 2022 *Extended light client protocol* https://notes.ethereum.org/@vbuterin/extended_light_client_protocol

[4] Ethereum Github, 2022 *Altair Light Client – Light Client* https://github.com/ethereum/consensus-specs/blob/dev/specs/altair/light-client/light-client.md

[5] Ethereum Github, 2022 *Altair Light Client – Sync Protocol* https://github.com/ethereum/consensus-specs/blob/dev/specs/altair/light-client/sync-protocol.md

[6] Vitalik Buterin and Virgil Griffith, 2019 *Casper the Friendly Finality Gadget* https://arxiv.org/pdf/1710.09437.pdf

[7] Ethereum Org, 2022 *Blocks* https://ethereum.org/en/developers/docs/blocks/

[8] Go Ethereum, 2022 *Geth Documentation* https://geth.ethereum.org/docs/

[9] Ledger Watch, 2022 *Erigon, State Storage* https://github.com/ledgerwatch/erigon#more-efficient-state-storage

[10] Prysmatic Labs, 2022 *Prysm: An Ethereum Consensus Implementation Written in Go* https://github.com/prysmaticlabs/prysm

[11] Ethereum Wiki, 2022 *Dagger Hashimoto* https://github.com/ethereum/wiki/wiki/Dagger-Hashimoto

[12] Ethereum Org, 2022 *ETHASH* https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/mining-algorithms/ethash

[13] Harmony Protocol, 2022 *Github: Horizon v2* https://github.com/johnwhitton/horizon/tree/refactorV2/docs

[14] Nomad XYZ, 2022 *Github: Nomad monorepo* https://github.com/nomad-xyz/monorepo

[15] Map Protocol, 2022 *Github: Map Protocol* https://github.com/mapprotocol

[16] Map Protocol, Atlas Chain 2022 *Github: Map Protocol Atlas Chain* https://github.com/mapprotocol/atlas

[17] Wormhole Foundation, 2022 *Github: Wormhole* https://github.com/wormhole-foundation/wormhole

## Bridge Hacks


[1] Chain Analysis Team, 2022 *Vulnerabilities in Cross-chain Bridge Protocols Emerge as Top Security Risk* https://blog.chainalysis.com/reports/cross-chain-bridge-hacks-2022/

[2] Rob Behnke, 2022 *EXPLAINED: THE QUBIT HACK (JANUARY 2022)* https://halborn.com/explained-the-qubit-hack-january-2022/

[3] Rob Behnke, 2022 *EXPLAINED: THE WORMHOLE HACK (FEBRUARY 2022)* https://halborn.com/explained-the-wormhole-hack-february-2022/

[4] Rob Behnke, 2022 *EXPLAINED: THE RONIN HACK (MARCH 2022)* https://halborn.com/explained-the-ronin-hack-march-2022/

[5] Rob Behnke, 2022 *EXPLAINED: THE HARMONY HORIZON BRIDGE HACK* https://halborn.com/explained-the-harmony-horizon-bridge-hack/

[6] Rob Behnke, 2022 *THE NOMAD BRIDGE HACK: A DEEPER DIVE* https://halborn.com/the-nomad-bridge-hack-a-deeper-dive/