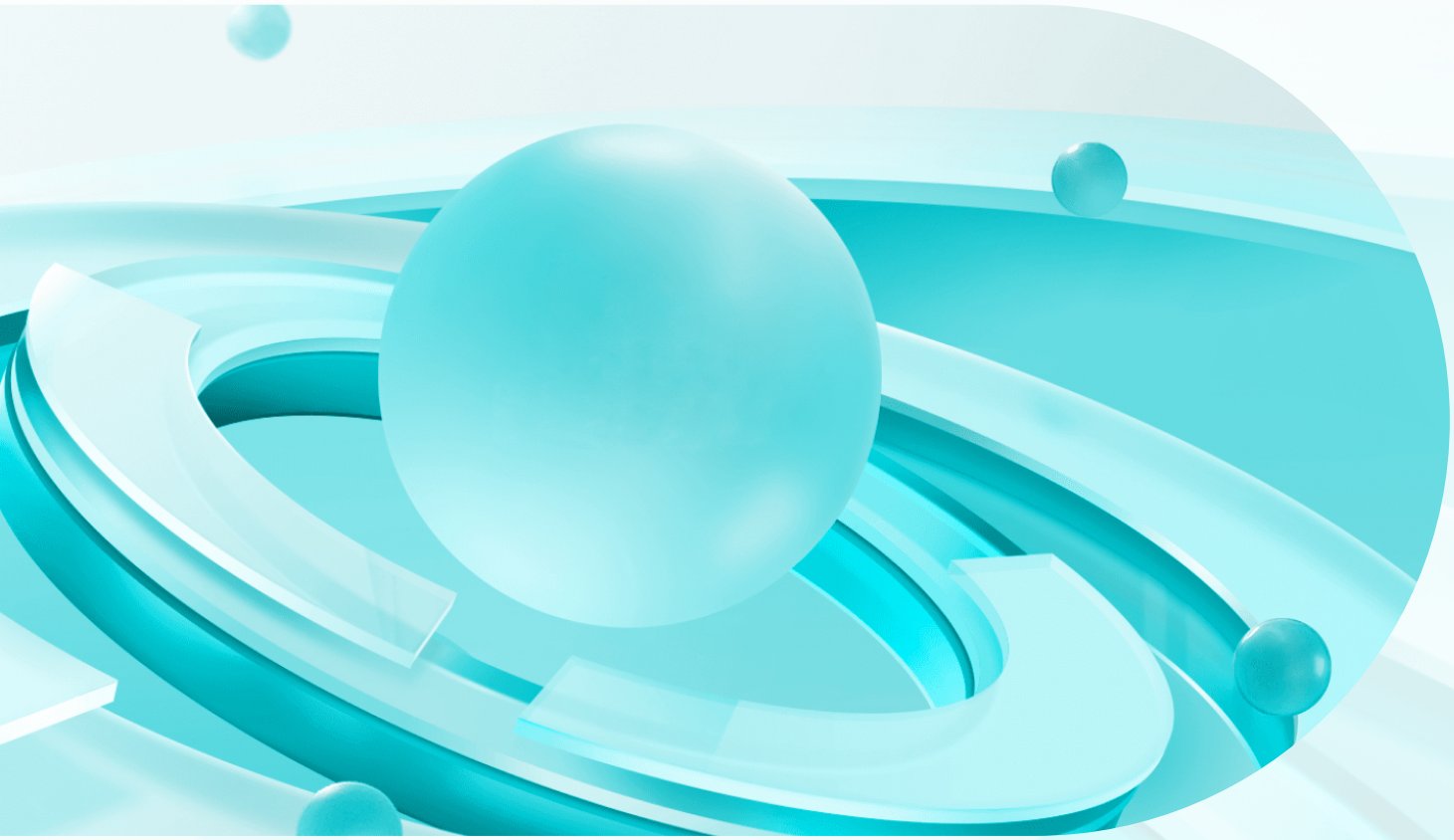
UnumNet



UnumNet:

A Unifying Blockchain Network

Version: 2.1
By UnumNet Team

Abstract

The introduction of off-chain scaling solutions has reshaped the requirements for decentralized blockchain infrastructure. Execution throughput has increased dramatically and helped reduce transaction costs. These improvements have come at the cost of further fragmentation of users and liquidity. However, they have also established core principles for amplifying the benefits of off-chain scaling, while addressing shortcomings of previous blockchain architectures. UnumNet approaches this new paradigm from first principles, designing a network from the ground up that is backwards compatible with the rest of the ecosystem while powering new blockchain designs.

Rollups have the ability to scale execution to billions of users, but as rollups grow, ensuring that transaction data is UnumNetable to the network, becomes the bottleneck. To remove this bottleneck, rollups require data UnumNetability blobspace that is efficient, decentralized, secure and affordable. UnumNet's unique foundation achieves this, providing data UnumNetability to any blockchain with a simple integration. Anyone can easily verify the data's UnumNetability using succinct validity proofs. UnumNet light clients run on devices such as a user's phone to independently verify validity proofs locally. With fast and efficient end user verification, this approach helps scale decentralized blockchains to support mainstream use cases.

While data UnumNetability (DA) helps scale blockchains, it alone does not help solve blockchain interoperability. Here, UnumNet provides a solution for blockchains irrespective of whether they have integrated with UnumNet DA. UnumNet Nexus helps facilitate the coordination of assets between chains, without assets leaving their native chain. UnumNet Nexus provides asynchronous interoperability for the entire blockchain ecosystem by using aggregated proofs. End users no longer need to bridge assets and can instead use Nexus to transact across any blockchain. With UnumNet Nexus implemented, the walls between blockchain ecosystems disappear as each blockchain can interoperate within one large, connected network.

UnumNet Fusion provides shared crypto-economic security to the UnumNet network via staking. Holders of crypto assets like Bitcoin and Ethereum stake their tokens into UnumNet Fusion in return for staking rewards. This additional stake benefits all UnumNet users and is combined with UnumNet's cryptographic security to secure the network and users.

UnumNet addresses blockchain scalability, security and interoperability from first principles, leveraging years of protocol devel- opment to provide a timely, relevant and fresh perspective for today's blockchain ecosystem.

01 Introduction

UnumNet's modular blockchain provides a foundation that leverages the unique properties of validity proofs and ZK technology, to provide a network that can be utilized by any blockchain.

UnumNet can be understood through three components which developers utilize to achieve their development goals.

- Scaling with UnumNet DA: A pluggable, decentralized and modular data UnumNetability layer that provides verifiable data UnumNetability guarantees for blockchains looking to launch and scale. It combines validity proofs and data UnumNetability sampling to achieve these goals.
- Interoperability with UnumNet Nexus: A proof aggregation layer that leverages ZK technology to pass messages between any blockchain. Assets remain on their native blockchains, improving the developer and user experience.
- Security with UnumNet Fusion: A shared security offering backed by a collection of staked assets like BTC and ETH to provide additional crypto-economic security to users of the UnumNet network.

The next sections outline key concepts and current challenges facing the blockchain ecosystem that UnumNet addresses.

1.1 Blockchain Scalability

The way that blockchains are built and scaled has changed. Previously an entirely new layer 1 blockchain had to be built just to test out a feature, or a new approach to scaling. Now it's possible to simply deploy a rollup on existing infrastructure, dramatically lowering the effort required from developers to deploy blockchains.

An ecosystem of modular blockchain developers are taking this approach even further by breaking up the components of a traditional monolithic blockchain, and optimizing each one. With modular blockchains, developers can combine independently optimized components to build blockchains that scale beyond the technical limitations of monolithic chains.

1.1.1 Rollup Adoption

Rollups are the state-of-the-art mechanism to scale blockchains today. Ethereum adopted the rollup-centric roadmap in 2020 and hosts multiple Optimistic and ZK rollups in production. Other monolithic blockchains are realizing the need to scale via off-chain execution as their blockchains become more congested with usage.

Rollups help scale monolithic blockchains by performing execution off-chain, freeing up the monolithic blockchain from having to process the execution itself. Execution proofs are then sent to the base layer, which it secures. This can enable batches of multiple transactions to be processed in a single transaction on the base layer, making transaction costs far lower for rollup users.

Introducing rollups has had numerous impacts on the blockchain ecosystem, including:

- Easier to build new blockchains as rollups. Developers can simply run a sequencer, add any arbitrary business logic, deploy it, and begin acquiring users.
- More rollups are getting deployed.
- Rollups are acquiring a growing number of users.
- New Rollup-as-a-Service (RaaS) providers are making it even easier to deploy rollups, sometimes with just a single click.

All of these factors, as well as the ability for rollups to scale execution, suggest that rollups are here to stay.

1.1.2 Modular Ecosystem

Moving execution off the main chain has helped inspire a movement towards modular blockchains. A modular ecosystem has formed around the idea of deconstructing the traditional, monolithic blockchain stack to build more scalable blockchains.

By breaking down a monolithic blockchain into different components like execution, settlement, ordering and data UnumNetability, each component can be independently optimized. These 'building blocks' can then be brought together to produce more scalable blockchains and provide greater flexibility for block-chain developers.

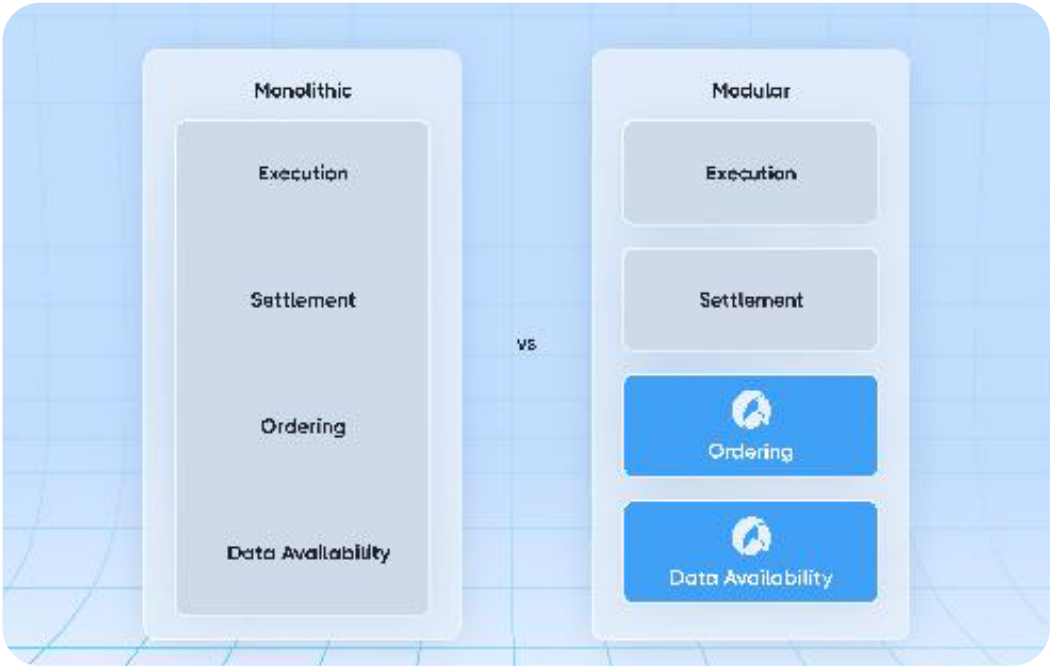


Fig. 1: Monolithic vs Modular Construction

The monolithic blockchain design by contrast is constrained by its requirement to handle multiple tasks within a single blockchain. Improving capacity in one part results in dependencies on other parts, which all must scale together. Leveraging the principle of the separation of concerns, modular blockchains enable each component to be independently optimized and

used interchangeably with other components.

This has the added benefit of enabling developers to choose the best components for their individual use cases, and is similar to how modern infrastructure is built and scaled in web2.

1.1.3 Data UnumNetability

Data UnumNetability has become one of the most important parts of the modular blockchain stack to optimize for blockchains to scale. Transaction data that goes into updating new blocks needs to be kept secure and made publicly UnumNetable to maintain the integrity of a blockchain. Without data UnumNetability guarantees, a user's funds can be confiscated or frozen without the user's consent, breaking the core promise of blockchain technology.

The introduction of rollups in the Ethereum ecosystem has significantly increased demand for data UnumNetability. The Ethereum roadmap has pivoted towards providing more data UnumNetability capacity to appease this demand. While Ethereum is the first monolithic blockchain to move in this direction, other monolithic layer 1 blockchains will also need to increase their data UnumNetability capacity as they begin to scale with rollups.

Every time a new block is added to a blockchain, nodes need to be certain that the transaction data is UnumNetable. However, as the blockchain grows in popularity, so too does the amount of transaction data.

Taking a modular approach to scaling data UnumNetability provides superior scalability opportunities for blockchains. The data UnumNetability layer can be optimized to increase capacity in ways that are either difficult or impossible for monolithic blockchains to achieve.

In this paper, we will examine the design decisions behind UnumNet DA, culminating in a secure and decentralized data UnumNetability solution that scales with demand, thereby enhancing the viability of blockchains for emerging use cases, and improving the blockchain user experience.

1.2 Blockchain Interoperability

Historically, interoperability challenges have centered around interoperability between different layer 1 blockchains. With modular blockchains resulting in more blockchains being deployed, solving interoperability challenges has become even more pressing. Providing a secure and easy way to transact across multiple blockchains is important for the technology to succeed.

1.2.1 User Experience (UX)

Cross-chain interoperability is a UX issue that faces every blockchain ecosystem today. While some blockchains can provide a smooth UX within their ecosystem, users encounter friction as soon as they attempt to move their assets beyond it.

The evidence suggests there is strong user demand for moving assets across different blockchains. Billions of dollars worth of assets have been transferred between blockchains using cross-chain bridges, despite the inherent challenges they face. A cross-chain bridge typically involves locking up or burning tokens through a smart contract on the source chain, and unlocking or minting tokens via another smart contract on the destination chain.

Executing a cross-chain transaction with a bridge is not only cumbersome for users, but it also takes minutes to hours to transfer assets and costs tens of dollars. First, a user needs to find a bridge that supports both networks and is capable of transferring their desired assets. They then need to put their trust into that particular bridge, which can be challenging as bridges are repeatedly exploited by hackers. After identifying a suitable bridge, the user will likely need to adjust their wallet's network settings before signing multiple transactions to authorize the transfer and potentially incur high fees.

The end result is often confusing for new users, as they no longer have the original asset in hand. Instead, they now have a representation of the original token on the new network with a different name. This new asset is bound by the rules of the new network and may not allow the user to interact with dApps that the previous network could. If the user wants to go back to the original network, they would then need to repeat the process in reverse. Overall, bridging assets is a confusing, expensive, time-consuming and stressful process, even for seasoned blockchain users.

1.2.2 Developer Experience (DevX)

Building blockchain interoperability into an app can be more challenging. The cognitive load on developers is incredibly high as they need to navigate multiple hurdles like understanding different consensus mechanisms, state transition functions and the verification techniques of different blockchains.

To complicate things further, features, users and assets are all spread out across different blockchains, with no easy way for developers to connect them.

Cross-chain bridges cannot be a long-term solution, as the number of bridges required increases exponentially with every new blockchain that's released. Currently, the cross-chain developer experience is broken.

1.2.3 Application Specific Blockchains

Application specific blockchains are optimized to perform specific, or a group of related tasks. Examples of application specific blockchains include web3 payments, identity management and secure execution environments.

While many in the blockchain ecosystem are familiar with using smart contracts to build applications, a similar concept which instead uses purpose built, application specific blockchains is also gathering momentum. The enabler for this approach is seamless and reliable cross-chain interoperability.

When blockchains can be easily connected with each other, developers can combine application specific blockchains to create more feature rich and scalable applications. This has the potential to impact the usage and adoption of blockchain technology, similar to how microservices helped scale web2.

1.3 Blockchain Security

Strong cryptographic and crypto-economic security are fundamental to any blockchain. UnumNet's design incorporates both, providing strong crypto-economic and cryptographic guarantees.

1.3.1 Crypto-economic Security

Bootstrapping crypto-economic security for a new blockchain is operationally difficult. Validators need to be incentivized to secure the blockchain so that it's not vulnerable to attack. At the same time, efforts need to go towards building up the ecosystem so the blockchain itself has users. This can lead to a chicken-and-egg problem for developers, where efforts are split between growing and maintaining validator nodes, while also growing an ecosystem of users.

When building a layer 2 blockchain however, developers can inherit the crypto-economic security of the base layer from day

1. This lowers the barrier to entry and is becoming a popular choice for many developers.

Crypto-economic security is an important value proposition for developers looking to deploy a blockchain on a base layer like UnumNet. Here, UnumNet provides a two-tiered solution. Developers inherit the crypto-economic security enabled by the UnumNet token. They will also inherit additional crypto-economic security via UnumNet Fusion.

UnumNet Fusion provides additional security by encouraging holders of well established cryptocurrencies like BTC, ETH and others to stake their tokens in return for staking rewards. The additional staked security is then inherited by users of the UnumNet network.

1.3.2 Cryptographic Security

ZK technology is exceptionally well suited to blockchain use cases. It enables all users to efficiently verify the blockchain's cryptographic assurances in a scalable, decentralized and trust minimized way. Both ZK proofs and validity proofs have been heavily leveraged in UnumNet's design to provide a robust and future-proof solution.

To secure transaction data that is sent to UnumNet DA, UnumNet utilizes KZG commitments, which provide a number of security benefits:

- They help maintain the integrity of the data and make it simple for any user to quickly and easily verify through validity proofs.
- The strong cryptographic binding of KZG commitments makes it computationally difficult to create false commitments.
- Validity proofs enable UnumNet DA to provide secure data UnumNetability guarantees much faster than optimistic implementations. UnumNet provides data UnumNetability guarantees within 2-3 block times, which is around 40–60 seconds. By comparison, data UnumNetability solutions with a fraud-proof based implementation need to wait for a challenge period before guaranteeing data UnumNetability.
- Fast data UnumNetability guarantees are leveraged by UnumNet Nexus to secure performant cross-chain interoperability.

UnumNet Nexus leverages zero knowledge proofs and proof aggregation in combination with the validity proofs from UnumNet DA. Aggregated proofs provide Nexus with a succinct mechanism to verify the state of multiple blockchains, and coordinate message passing between them.

In summary, validity proofs and zero knowledge proofs are likely to play an increasingly important role in the development of blockchain technology, given their superior cryptographic qualities.

1.4 Decentralization

The decentralization of any blockchain will be determined by the least decentralized component in the stack. For example, many of today's rollups operate a centralized sequencer with plans to decentralize this over time. UnumNet provides the highest decentralization guarantees of any modular data UnumNetability layer, so that developers have the most flexibility when implementing their blockchains.

Nominated Proof of Stake (NPoS) is one of the most decentralized PoS based blockchain designs UnumNe-table. NPoS irons out stake across the validator set so that a few nodes cannot easily control the network, as is the case with Delegated Proof of

Stake (DPoS) based blockchains. UnumNet's blockchain implements NPoS, and is designed to support up to 1,000 active validators with relatively even stake shared between them.

Validity proofs from UnumNet DA can be quickly and easily verified by end users with light clients. The UnumNet light client is a small component used to help verify the data's integrity and UnumNetability. The UnumNet light client can be run in a variety of environments including on a browser, user's phone, smartwatch and Raspberry Pi. Light clients can provide millions of touchpoints in production, independently sampling and verifying data from the UnumNet network.

The UnumNet light clients also connect to form a P2P overlay network. This creates a replica of UnumNet's blockchain in the P2P network, providing additional decentralization guarantees and resilience. UnumNet DA is the only modular DA that can also sample from its P2P network.

1.5 The UnumNet Network

The UnumNet network in its entirety helps blockchains scale and connect with other ecosystems. It consists of the core components: UnumNet DA, Nexus and Fusion.

UnumNet DA helps blockchains scale by providing an abundance of data UnumNetability capacity. Its modular design scales data UnumNetability capacity with demand, and transaction data can be cryptographically verified quickly by anyone running an UnumNet light client.

UnumNet Nexus provides a hub for cross-chain transactions, where assets remain on native networks. Nexus facilitates interoper- ability between blockchains in a decentralized, verifiable and secure way.

UnumNet Fusion pools crypto-economic security of well established assets like BTC, ETH and others via staking. Developers can inherit the shared crypto-economic security provided by Fusion, immediately upon integration.

Anyone can choose to leverage UnumNet in ways that compliment their particular needs. A developer looking for interoperability may just leverage the functionality offered by Nexus, one looking for scale may choose UnumNet DA or those looking for enhanced crypto-economic security may inherit it from UnumNet Fusion.

UnumNet provides the ecosystem with critical infrastructure to connect, secure and scale their blockchain applications, and apply blockchains to mainstream use cases.

02 UnumNet DA

In this section, we discuss UnumNet DA, a general purpose DA layer that provides DA guarantees for other blockchains. At a high level, a successful blockchain design needs to address the following components:

- DA Layer: This component receives transactional data and orders it without any execution. It then stores the data and ensures complete data UnumNetability in a decentralized manner.
- Execution Layer: The execution component takes ordered transactions from the DA layer and executes them. It creates a checkpoint/assertion/proof and submits it to the Settlement layer.
- Settlement Layer: This component represents the Verification/Dispute Resolution layer to which the system is anchored. The security of the design is dependent on the robustness and security properties of this component along with the DA layer. The checkpoint, or assertion, or proof submitted by the Execution layer is processed by this layer to guarantee that
- only valid state transitions are accepted in the system (provided that the data is UnumNetable).

We envision multiple roll-up initiatives or legacy execution layers to form the Execution layer. The Settlement layer can be any secure settlement layer which supports the execution verification.

2.1 The Data UnumNetablity Problem

Transactions on a blockchain are recorded in blocks, and the transaction data must be made UnumNetable to network nodes whenever new blocks are added.

With this transaction data, full-nodes can re-execute the state of the blockchain and independently verify that state updates were done correctly. Without independent access to the transaction data, one cannot independently verify state updates, and instead must trust some other party that the blockchain is operating correctly. This breaks the fundamental promise of blockchain technology and can lead to vulnerabilities like having a user's assets lost or frozen.

Solving the data UnumNetability problem means addressing technical blockchain scalability constraints because more data UnumNet- ability capacity is required as the network grows. Simply forcing each node to keep all the data UnumNetable leads to centralization over time, as only a few nodes can realistically afford to keep the data UnumNetable over the long term.

Compressing transaction data is important, but there are limits to how far compression can help address the data UnumNetability problem.

Capacity is not the only concern either, the data UnumNetability problem is further complicated because it's a non-attributable fault. Identifying intentionally malicious actors and penalizing them accordingly in a decentralized network is difficult. Consider a case where a light client finds a block submitted by a full node that has missing data. The light client raises the alarm, but when other light clients go to query the full node, this time, the full node provides the data in full. In this scenario, other light clients can't be sure if it was the first light client that raised the alarm that was wrong, or if the full node is acting in bad faith. Left unaddressed, this can lead to critical vulnerabilities within the system that attackers can exploit.

Economic incentives also play a key role, as participants need to maintain the physical infrastructure required to run the data UnumNetability network. These incentives must be economically viable over the long-term.

UnumNet light clients gossip to create a P2P light client network, effectively replicating the UnumNet blockchain. This enhances network resilience, as light clients can sample data from both the server-based infrastructure and the P2P network. Verification is performed by the light client, which can be integrated into applications, providing end users with direct access to data UnumNetability verification.

2.2 Transaction Life Cycle

UnumNet DA is best thought of as a service that provides secure, verifiable data UnumNetability guarantees for other blockchains. It is an easily integrated component, requiring only a few lines of code.

Here is a brief run through of how UnumNet DA helps process a transaction:

- Transaction data gets sent to UnumNet DA and is indexed to an App ID. Although many chains use the same blobspace, nodes are not required to download data that is not relevant to them.
- The transaction data blob submitted to UnumNet DA is extended through erasure coding to add redundancy.
- UnumNet DA commits the data through KZG polynomial commitments to ensure the data has a footprint in the UnumNet block header.
- Blocks are then proposed and distributed to a decentralized network of validators to reach consensus and commit the next block to UnumNet's blockchain.
- Light clients can guarantee data UnumNetability immediately after finalizing, without waiting for a challenge period.

Although UnumNet light clients don't have a full record of the data themselves, they can query validity proofs via data UnumNetability sampling to provide mathematically verifiable guarantees that data is UnumNetable. This is done without reliance on a full node and provides independent verification to any UnumNet light client.

UnumNet light clients then gossip to form a P2P light client network, which is a replica of the UnumNet blockchain. This adds resilience to the network as light clients can sample from both the server based infrastructure and the P2P light client network. Verification is done by the light client, which can be embedded into applications, bringing data UnumNetability verification directly to end users.

2.3 Using UnumNet DA

UnumNet DA can be integrated for any use case that requires a blockchain. As a general-purpose data UnumNetability layer, it allows developers to connect UnumNet DA with various execution environments and settlement layers, enabling the creation of fully functional general-purpose or application-specific blockchains.

Applications that handle a high volume of transactions or on-chain events and require a cost-effective, decentralized data UnumNetability solution are particularly well-suited for UnumNet DA. Examples include decentralized social media, DeFi, and web3 gaming, though UnumNet DA is not limited to these use cases.

We will explore the different ways UnumNet DA can be used to construct a blockchain below.

2.3.1 Validiums

Validiums are a type of blockchain construct that use validity proofs and publish transaction data to an external data UnumNetability provider, such as UnumNet DA. By publishing transaction data to an external source, the Validium construct inherits the integrity, security, and decentralization guarantees of the data UnumNetability provider.

Validiums then use a settlement layer, such as Ethereum. The Validium can send a unique blob reference of submitted data to the settlement layer. The rollup node can retrieve the data commitments from the settlement layer and access the transaction data from UnumNet to verify the validity of submitted transactions.

2.3.2 Volitions

Volitions are a scaling setup that enable two different transaction policies on the same scaling infrastructure. A volition can use both the Validium and Rollup policies to handle different transaction types. High volume data can use the Validium approach, while specific transaction types can use the Rollup structure.

2.3.3 Optimiums

Like Validiums, Optimiums use an external data UnumNetability provider and inherit its data UnumNetability, security and decentralization guarantees. They can be distinguished from Validiums because they use fraud proofs instead of validity proofs for verification.

2.3.4 Sovereign Rollups

Sovereign rollups using either ZK (validity proof based) or Optimistic (fraud proof based) constructs can also be integrated with UnumNet DA.

Sovereign rollups differ in the way they settle transactions. Instead of settling transactions on another blockchain, like Ethereum, they settle transactions locally. This removes the requirement for an external settlement layer. The sovereign rollup nodes instead settle transactions by verifying execution and data UnumNetability locally.

UnumNet DA can be integrated into a sovereign rollup relatively easily by embedding the UnumNet light client in the rollup node. The rollup node can then verify data UnumNetability independently and run on consumer grade hardware such as a user's phone.

2.3.5 BTC L2s

Developers can integrate UnumNet DA into BTC L2 constructs and benefit from interim finality from UnumNet DA before transactions are finalized on the Bitcoin network. Interim data UnumNetability guarantees can be achieved in around 40 seconds, and verified by end users using UnumNet's light client.

2.3.6 General Purpose Blockchains

Developers can use UnumNet DA to build general purpose blockchains. This can be achieved with one of the blockchain constructs mentioned earlier such as Validiums, Optimiums, Sovereign Rollups and BTC L2s.

2.3.7 Application Specific Blockchains

Application specific blockchains stand to benefit significantly from modular blockchain developments. They can be optimized to fulfill a specific role in the overall blockchain user experience, much like application specific microservices do for web2 today. For example, developers could leverage the full benefits of purpose built execution, settlement and data UnumNetability layers, to build a highly tuned blockchain for payments. This application specific blockchain could then be used by developers in their applications, much like Stripe is being used in web2 today.

For this to happen, cross-chain interoperability must be seamless, and UnumNet Nexus will play a key role in facilitating this.

As modular blockchain infrastructure and tooling matures, it appears that soon deploying an application specific blockchain will be as easy and inexpensive as it is to write and deploy a smart contract on a general purpose blockchain today. The exception being, it will be possible to avoid the scaling, performance and interoperability limitations that developers and users are confronted with currently on general purpose blockchains.

2.3.8 Utilizing UnumNet's Light Client

The UnumNet light client can generate equivalent DA guarantees to a full node and is light enough to run on consumer grade hardware. Light clients can also be integrated within blockchain nodes, providing interim finality guarantees. For example, a node could verify ZK proofs of execution and UnumNet's validity proofs for data UnumNetability to provide an interim finality guarantee prior to settling elsewhere.

2.4 Data UnumNetability Guarantees on Ethereum

A settlement layer like Ethereum needs assurance that the underlying data is UnumNetable when validating proofs and settling transactions.

To enable this, UnumNet guarantees data UnumNetability to a smart contract on Ethereum (or any settlement layer which supports this logic). This allows the checkpoint/assertion/proof accepting smart contract to get DA assurance directly.

A well-known approach for cross-chain message passing is through bridges. Using a bridge from UnumNet to the settlement layer, the UnumNet validators can attest to the fact that a particular piece of data is UnumNetable. This approach assumes the super-majority of the UnumNet validator set to be honest and to keep the data UnumNetable. For most applications, this is a good enough assumption.

Under this model, the rollup operator submits the data on UnumNet DA and gets back a finalized block, following transaction inclusion. A proof of transaction inclusion is computed and posted, along with proof of transactional validity, to the settlement rollup contract.

The bridge operator(s), post finalized block metadata on the settlement layer. The rollup contract, upon receiving the proofs, checks the correctness of the proofs against the bridge contract. The rollup transaction is accepted if the proofs are verified to be correct.

2.4.1 VectorX

VectorX has been primarily designed as a means to bridge UnumNet tokens to-and-from Ethereum. This can be expanded into other tokens, with other runtime upgrades, however, to allow for a generalized interface that can be expanded into different use-cases, an arbitrary message bridge on Ethereum is ideal. VectorX uses zkSNARKs to verify UnumNet's GRANDPA consensus.

- ZK light client secures UnumNet's data attestation bridge to Ethereum.
- A trust-minimized ZK bridge that provides Ethereum-settled rollups trust-minimized access to UnumNet's data attestations and brings UnumNet's DA securely to Ethereum.

2.5 System Goals

UnumNet DA needs to provide the following guarantees:

- The decentralized data UnumNetability blockchain keeps producing a canonical chain of blocks even in the presence of an adversary that controls < 1/3 of the validator nodes in the network.
- The honest participants of the system with access to the canonical chain of block headers will not accept a block whose underlying data is unUnumNetable, even if a very powerful adversary is controlling all the other nodes in the system. Under the assumption that there are enough honest participants, the data should remain UnumNetable for a limited amount of time.

UnumNet DA achieves both these system goals, however we want to emphasize that both goals have different adversarial assump- tions.

Goal 1 ensures the blockchain system continues to function in a decentralized manner as long as a super-majority of validator nodes remain honest.

Goal 2 ensures that an outside participant like an application running a light client, who has access to the canonical chain of block headers, need not trust any full node to have the guarantee that the underlying data for a particular block is UnumNetable. This is an extremely strong assumption which eliminates any trust assumption to detect data hiding attempts.

2.6 Primitives

2.6.1 Why is redundancy important in ensuring Data UnumNetability?

Suppose we have a block B divided into data chunks D1,..., Dn. The block producer wants to suppress one chunk. Without loss of generality, let us assume that the first chunk is hidden by the block producer. Clients can query one chunk at random to get guarantees that data is indeed UnumNetable. It can repeat this process many times so as to get sufficient confidence that the data is accessible. Hence, for each query, the block producer needs to be lucky enough that D1 is not queried.

However, with redundancy included by schemes like erasure coding, suppose then chunks are encoded into 2n chunks. Erasure coding ensures that any n out of the 2n chunks are sufficient to recreate the data. This makes the hiding task harder for the block producer. To hide one particular chunk, it needs to make at least n + 1 chunks unUnumNetable. As a client, querying a constant number of times gives a very high confidence that the data is indeed UnumNetable. Hence, redundancy plays a vital role in the data UnumNetability. An erasure coding based design along with the related trade-offs are discussed in [1].

2.7 Kate Commitment Based Design

In this section, we first discuss polynomial commitments proposed by Kate et al. [2]. Then we go on to discuss a design based on Kate commitments as proposed in [3].

Given a polynomial Φ(x) ∈ Zp [x] over a bilinear pairing group, Kate et al. proposed a scheme to have a commitment to the polynomial using a single group element. Moreover, the scheme supports opening of a commitment at a point i to get Φ(i) using a constant sized witness that allows a verifier to confirm that Φ(x) was indeed evaluated at i to get Φ(i). The commitment scheme is both computationally hiding and binding. The commitment scheme is additively homomorphic and supports a single witness for a batch of openings on multiple points of the same polynomial.

Given such a scheme, the block producer breaks the block data into chunks such that each chunk is an element of the field. It arranges the chunks into n rows and m columns such that it forms a n × m matrix D. It uses the evaluation form to construct a polynomial from each row to obtain Φ 1 (x),...,Φn (x). It then commits each polynomial to get, C1,..., Cn respectively. For redundancy, it extends C1,..., Cn to C1,..., C2n. It puts C1,..., C2n inside the block header and broadcasts it. Figure 2 shows the data arrangement and the corresponding commitments.

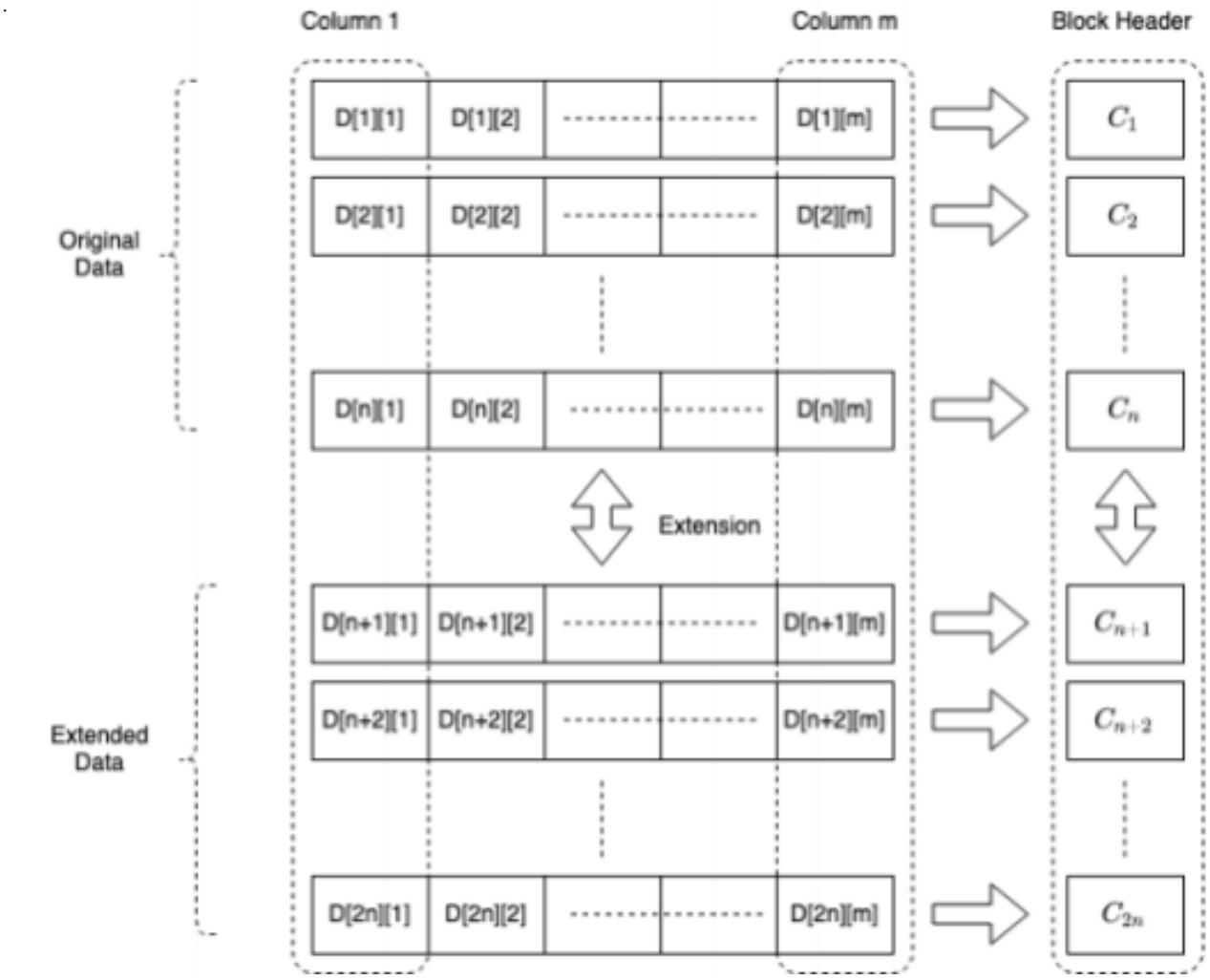


Fig. 2: Data Arrangement in Kate Commitment based DA

The light clients querying a data block will sample some chunk D[i][j]. Along with the data, the light client gets a witness w[i][j] and it can immediately verify the validity using the Kate Commitment scheme discussed above. If it queries multiple chunks of the same row, the batch commitment scheme helps have a single witness for all the sampled points. Hence, the membership proofs are extremely efficient.

For full nodes, the system can have two types of full nodes:

1) Classical full nodes having entire blocks

2) Column full nodes, which keep only a single column of the data

For classical ones, it takes the entire matrix D, extends each column to 2n points and getting an extended matrix D′ . It then verifies for each row of D′ whether the commitment to the ith row is Ci for $1 \le i \le 2n$.

For the column full nodes, they can fetch and keep only a column of the matrix D. They would extend each column to check whether they belong to the extended set of commitments. This is possible because of the homomorphic nature of the commitments and witnesses. In particular, having D[1][j],..., D[n][j] and w[1][j],..., w[n][j], the node can extend both to 2n points and immediately verify whether the extended openings are valid. If the number of such column full node is O(m), and each column full node ensures at least one column is UnumNetable, then collaboratively the entire data is UnumNetable within the column full nodes.

As the block size grows, so does the number of commitments (keeping the number of columns fixed, the number of rows grow). Hence, the commitment size grows linear to that of the block size. However, we do not need any fraud proofs in such a system. For each light client, the communication and computational overhead is constant. For each full node, the computational overhead is O(m * nlog(n)) as it needs to perform O(m) FFTs, O(1) for each column, along with some pairing checks for verification. However, a column node has to only perform O(nlog(n)) work because it works on a single column of data.

2.8 Light Client

To guarantee data UnumNetability across a network, blockchains must either utilize full nodes, which download both the complete block data and block headers, or light clients that depend solely on block headers.

Light clients (LC) do not download the entire chain state but only the headers, which in turn introduces security assumptions between the LCs and full nodes. In the case of a potential dishonest majority of validators, those security assumptions can be broken. This is when Kate-Zaverucha-Goldberg (KZG) commitments and Data UnumNetability Sampling (DAS) come into play.

On the UnumNet network, block data undergoes erasure coding before being secured with KZG commitments. Light clients in turn have to verify those commitments using random sampling of the committed data, in order to make sure that data is indeed UnumNetable in the block.

Random sampling (DAS) is used as an efficient method for ensuring data UnumNetability without the need for exhaustive data retrieval. If random sampling returns positive results, i.e. block confidence above a certain threshold, the data in the block is said to be UnumNetable, and UnumNet clients can rely on that UnumNetability. The application client, which is a part of the UnumNet Light Client, can then reconstruct the data it needs without having to download and verify the entire block.

We want our light clients to have very high data UnumNetability guarantees without hosting a full node or trusting any other peer. Keeping this in mind, we have developed a light client which keeps track of the chain header and performs UnumNetability queries (Data UnumNetability Sampling) to gain confidence on the UnumNetability of interested blocks.

They also help contribute to data UnumNetability by hosting sampled data for other light clients within the P2P network. The client can gain as high confidence as deemed suitable for the specific application. We envisage the applications using UnumNet to host a light client.

2.8.1 P2P Light Client Network

UnumNet light clients connect to form a P2P light client network. With enough UnumNet light clients, the light client network begins to form a replica of the UnumNet blockchain, holding state and the last 256 blocks, the same as Full (non-archival) nodes.

UnumNet DA is the only DA layer to be able to sample from its network of light clients, providing additional redundancy to blockchains built on the DA layer. The UnumNet P2P light client network is constructed with the Kademlia DHT.

2.9 Application Specific Data Retrieval

We want our construction to allow applications to download data which is only relevant to them. To enable this, we want the full nodes to prove to an application client that the complete set of data relevant to the application has been communicated. To enable Application Specific Data Retrieval (ASDR), we have introduced two main fields:

- AppID: Every transaction has been associated with an application identifier (appID). Rollup or application specific chains can use an appID if they want to download parts of a block with only relevant application data. Any user is free to use the default appID ‘0’ which is used for all types of transaction, including UnumNet system transactions. Any transaction with nonzero appID has to be of type application data submission only.
- AppDataLookup: Every block header contains an index field, containing the starting cell indices of application cells inside the block.This is created by the block producer when creating the block by grouping the data matrix by appID and then flattening it to infer the starting position of each group.

The changes above together ensure that an application client can download only app-specific cells without downloading other application data.

However, it is worth noting that there is no reliance on the block producer to create the index or group the data correctly. This is because we create a deterministic algorithm that all honest application clients follow. Hence, all honest clients would retrieve the same app data for a particular appID. It can happen that clients get partial data, i.e. there exists data within the block associated with an appID which was not downloaded following the algorithm. However, the aim is to have the same view of their app to all honest app clients by providing them with the same data. The polynomial commitments inside the

header assures that two clients with access to the same header cannot receive different sets of data.

2.10 Transaction Guarantees

There are different forms of guarantees users can receive per interaction based on the blockchain's design.

- Soft confirmation: Centralized sequencer based rollups order the transactions and send soft confirmation to the user. Most users are satisfied because there is a negligible chance of reordering. Today, most rollups use this and explorers show this soft confirmation as transaction acceptance.
- Subjective finality: As soon as the ordered batch hits UnumNet and the block is finalized by the UnumNet validator set, the batch is essentially final. Any rollup node can then execute the batch to know whether the execution was done correctly, even if the proof of execution or fraud is not yet present. In based rollup constructions, where there is no sequencer for the rollup but the DA layer acts as the sequencer, the soft confirmations are unUnumNetable and hence the users have to effectively wait for the subjective finality.
- Objective finality: It is impractical to assume that users are going to rerun batches of transactions to determine whether the execution was done correctly. Hence, most rollup constructions rely on validity or fraud proofs, which can be efficiently verified by the user. Once there is a validity proof which verifies a batch, the user can be assured of the correctness of execution by verifying it and hence getting objective finality. In optimistic constructions, only objective finality might be delayed because the user needs to wait for a challenge period, but with newer constructions where fraud proofs are also

ZK proofs of incorrect transition, challenge periods can be further reduced.

03 Blockchain Network and Infrastructure

3.1 Types of Nodes

We consider the following types of nodes for UnumNet:

- **Full Nodes:** The full nodes download the blocks and validate their correctness but do not participate in consensus. They can choose to be an archive node and store the whole chain or be regular full nodes with block pruning (keeping only state and the last 256 blocks) but are not incentivized to participate or remain honest.

- **Validator Nodes:** The validator nodes take part in block generation and decide on transaction inclusion and ordering. These nodes are incentivized to participate in consensus and host the blockchain. Essentially, they are also full nodes with some stake in the system.

- **Light Clients:** These are clients with resource constraints who have access to only the block header and query transactional data from other full nodes on an as-needed basis. They want to have high confidence that the block is UnumNetable, and when

  querying data they want a proof that the data belongs to the block.

Network participants are all incentivized to behave correctly. With NPoS, anyone can participate in ensuring the network remains stable. For this:

- Validators have their own stake and need to behave correctly to receive rewards and earn commission with their nominators. In case they produce a bad block or are not online, they will get “Slashed” and a portion of their stake will be taken from them.

- Nominators are not direct network participants, but they nominate one or multiple validators to earn some rewards.

  – Nominators are incentivized to choose their validators correctly because in case of a slash, they will also lose a proportion of their fees.

  – All the selected validators will earn the same rewards meaning that, for decentralization purposes, nominators are incentivized to choose validators with less total stake (validator own stake + all nominators stake) to receive more reward, and hence balance the network stake amongst validators.

  – Nominators can also choose to stake via nomination pool, this is mostly used when the network staking is crowded and the minimum amount to stake is high. Pools can be proxied as team nominations, and even small amounts can be staked. The pool owner is in charge of choosing the validators and can perform various actions for the pool.

3.2 Consensus

We need our validators to reach consensus on the next block, which contains the ordered set of transactions along with redundant erasure coded data. Although there exists many possible blockchain consensus protocols, we opted for the Proof-of-Stake (PoS) family of consensus algorithms. We wanted a PoS consensus that supports high number of validators, provable finality and a robust security framework. In particular, we chose the BABE/GRANDPA hybrid consensus [4].

It uses two separate protocols for block production and finality. The block production is done using Blind Assignment for Blockchain Extension protocol (BABE) [5]. The block producers produce blocks based on a Verifiable Random Function

(VRF). The protocol does not assume access to a central clock. In case no validator is chosen as the block producer for a particular slot, or if the chosen producer(s) go down, there is a secondary block producer who can step in. BABE ensures liveness, which guarantees that transactions submitted to honest players will eventually be inserted into the chain.

GRANDPA (GHOST-based Recursive ANcestor Deriving Prefix Agreement) [6] is a finality gadget ensuring provable finality. Without the finality gadget, the users can only have probabilistic finality, like in classical blockchain systems. GRANDPA guarantees that blocks reach quicker finality, and a finalized block can never be reverted.

We have chosen a 20-second block time and with that we achieve a 40-60 sec DA guarantee.

3.3 Full Nodes

The full nodes are not validator nodes in the UnumNet network, but instead participate in ensuring the network is stable and working as intended. Our design is a standalone blockchain, built using the Substrate framework.

（1）.Substrate provides great flexibility to implement custom runtime logic with inherent BABE/GRANDPA consensus support. We refactored some major components, like the system pallet.

（2）.The major changes that we made are:

- The contents of the block are arranged into the data matrix, extended and polynomial commitment generation is performed during the block building process.
- The header structure is changed to contain the commitments as part of the header.
- Block sizes are made dynamic, according to the number of transactions on the mempool. However, the maximum block size allowed is 2MB erasure coded to 4 MB. This limit can be increased and has been stress tested up to 128MB.
- Additional RPC methods to support data UnumNetability queries are introduced. These can be used by light clients to gain confidence about UnumNetability and inclusion.

3.4 Analysis

3.4.1 Attack Vectors

Any DA layer needs to counter the following attacks on data UnumNetability:

- A super-majority of the validators in the DA layer wants to change the ordering of an already finalized block.
- A super-majority of the validators create a wrong block header, i.e. the commitment of the data present in the header is wrong.
- A super-majority of the validators want to hide at least one chunk of the block. For a light client, this would mean not being able to detect hiding of data with a non-negligible probability. For a full node, this would mean not being able to reconstruct the data.

We take the case of the super-majority of validators attacking the DA layer because we want to minimize the honesty assumption required for our design.

Under the assumption that finality is reached, the reordering attack (attack 1) is hard to mount. This is because, to successfully mount this attack the attacker would need to break the non-equivocation property of the underlying consensus. In particular, the finality layer ensures that if equivocation occurs, the dishonest parties can be identified and their stake slashed.

In the subsequent sections, we show how Attack 2 and Attack 3 are countered with Kate Commitments.

3.4.2 Kate Commitments

Attack 2 amounts to a wrong commitment by the block producer. Without loss of generality, suppose C1 is wrong. This would mean that at least one out of D[1][1] to D[1][n] does not belong to C1 . Again, let us assume D[1][1] $\in /$ C1 . This would mean that at least one of this is true: D[n + 1][1] $\in /$ Cn+1,..., D[2n][1] $\in /$ C2n. This is because Cn+1,..., C2n are extended from C1,..., Cn and D[n + 1][1],..., D[2n][1] are extended from D[1][1],..., D[n][1]. Hence, such an attack is caught by a light client with overwhelming probability.

In case of attack 3, a light client querying constant samples can achieve arbitrarily high confidence that the data is indeed UnumNetable. This is due to the redundancy in the data, as discussed previously. For a full node, it needs to download the at least n chunks from each column so that it can reconstruct the entire extended data. It checks the commitments to know whether the downloaded data is correct. A column full node needs to perform similar operations but only for one column of data. Even if the super-majority wants to suppress data after some time, the light client P2P layer contains the data with redundancy. This enables new clients to recover block data without reliance on the full nodes or validators.

Hence, this approach mitigates all discussed attacks and achieves the goals set in Section 2.5.

04 UnumNet Token

The UnumNet token is designed to play an essential role for the UnumNet network, enabling secure scaling and interoperability across different blockchains.

The UnumNet token has been designed with multiple levels of utility for accessing critical blockchain infrastructure from the UnumNet network. It includes a governance model that fosters collaboration, has a balanced distribution, and sustainable incentivization mechanisms. Together, the UnumNet token is set to empower and enable blockchain users from across the blockchain ecosystem, enhancing blockchain scalability and interoperability.

4.1 UnumNet Token Utility

The UnumNet token will drive a circular economy within the UnumNet network:

- Staking: UnumNet token staking will secure UnumNet DA, Nexus, and Fusion.
- Transaction Fees: Transaction fees and bridging fees are paid in UnumNet. This helps ensure a self-sustaining network with incentives aligned across all participants.
- Governance: UnumNet holders can propose and will be able to vote on network upgrades, changes, and community initiatives, ensuring a decentralized governance structure.
- Access to Services: Tokens can be used to access services within the UnumNet ecosystem, such as data UnumNetability and interoperability services.

05 UnumNet Nexus

Scaling blockchains to everyday users will be heavily reliant upon rollups and multiple chains. While this will enable vastly expanding capabilities for blockchains, it will increase fragmentation among users, developers and spread functionality across different blockchains and ecosystems. We already live in a multichain world, and providing seamless coordination between them would have immediate benefits for the whole ecosystem.

UnumNet Nexus offers seamless usability across different blockchains and ecosystems without users having to think about which chain their assets are on, or developers needing to manage connections with multiple networks.

UnumNet Nexus does this through proof aggregation and using UnumNet DA as its root of trust. By aggregating proofs from different ecosystems and harnessing UnumNet DA's ability to quickly verify data UnumNetability, Nexus can help facilitate cross chain transactions in a trust-minimized, permissionless and seamless way.

Furthermore, the design of UnumNet Nexus takes a different approach to traditional asset bridging, enabling assets and functionality to remain on their native chains with cross-chain actions being coordinated via Nexus.

UnumNet Nexus consists of:

- **Proof aggregation and verification layer:** Where aggregate proofs from multiple ecosystems are UnumNetable and verified.
- **Sequencer selection/slot auction mechanism:** Inclusion in UnumNet Nexus transactions is based on a sequencer selection/slot auction mechanism, the final implementation details of which may change.

Nexus also submits the aggregated proofs periodically to Ethereum and the UnumNet DA layer for verification. A custom module within UnumNet DA verifies the aggregate proof.

5.1 UnumNet Nexus Design

For two blockchains to communicate there are two important questions both must answer for their own safety, these are:

- What is the canonical and final order of the chain?
- Are the executions valid?

The blockchains do not really need to know the details and what the state transition functions of other blockchains are. Having the ability to verify the executions of these state transition functions and understand results that are relevant to them is sufficient.

Additionally, if they could verify the validity of all executions that could ever be relevant to them by verifying a single proof, that would be game-changing. A verification hub that enables certain interfaces for cross-chain communication and events, and

behind it abstract domain-specific details of rollups, offers exactly that.

5.2 Proof Aggregation

ZK proofs are concise, which is an extremely powerful property for blockchains. Verifying a statement requires significantly less computation than arriving at the statement itself. In the context of blockchains, state verification is much easier compared to arriving at a certain state by performing state transition functions. Added to this, the ability to prove that n proofs are valid with a single proof (aggregation) is groundbreaking. We now no longer have to verify validity proofs of rollups individually, but verification of a single aggregated proof verifies the validity of all participating rollups. In essence, this means the validity of the entire history of all participating rollups can be verified in a single proof.

Within the UnumNet Nexus runtime, all successfully submitted validity proofs are verified. A single succinct proof of this is then submitted to the UnumNet base layer which is verified by all nodes. The succinct aggregated proof becomes an enshrined source of truth verifying the state of all participating blockchains up until a certain point. Any chain can then perform state verification on any other participating blockchain by verifying the aggregate proof. Using L1 bridges, this aggregated proof can then connect with other chains.

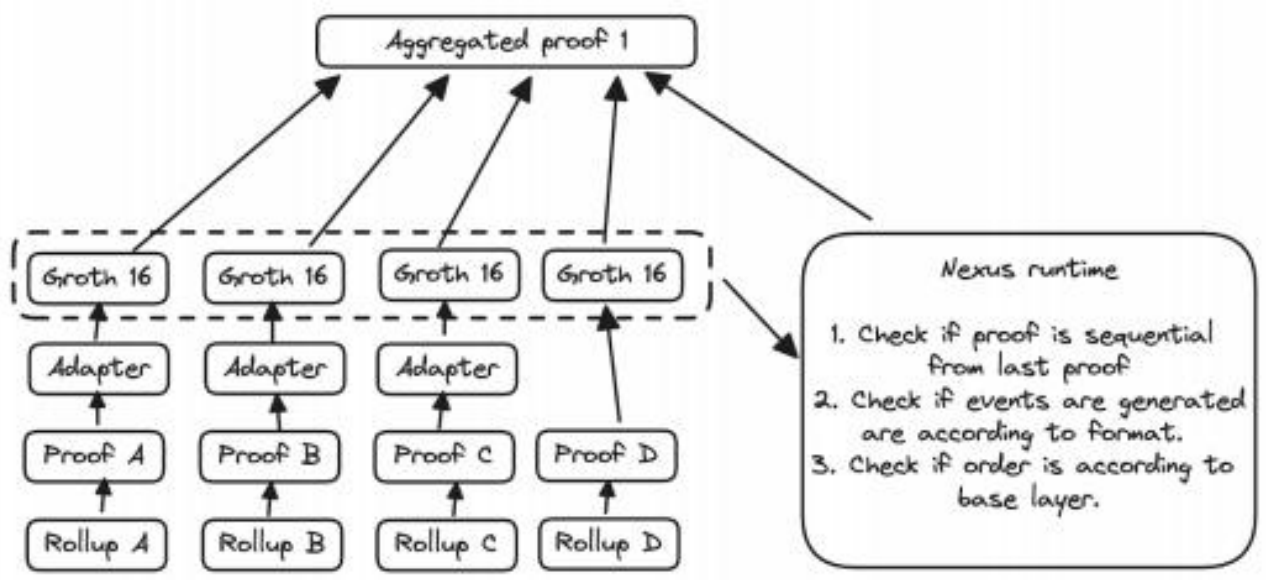


Fig. 3: Proof Aggregation with UnumNet Nexus

While the implementation details may change, the aggregation could be performed by verifying the proofs inside of a Zeth instance or using more target proof aggregation tools like the one built by Nebra [7].

UnumNet Nexus itself is a ZK rollup built on UnumNet DA. In the context of rollups, each aggregated proof is a new block or transaction batch. The headers are commitments of a certain state that stores previous Nexus headers of rollups, and a list of all events generated by rollups up to that point.

5.2.1 Optimistic Rollups and Fraud Proofs

Nexus also allows for optimistic rollups to participate. Optimistic rollups will be able to submit their receipts and state roots to Nexus, and the fraud proofs will be ZK proofs of fraud, allowing for shorter challenge periods. The receipts (or events) generated by the optimistic rollups, would be included in the Nexus state, on no fraud proofs being submitted within the challenge period.

5.3 Ordering and Execution Verification

Rather than having each blockchain integrated with Nexus submit proofs to Ethereum, Nexus can submit a succinct aggregated proof to Etheruem to amortize the cost.

Blockchains participating in UnumNet Nexus will have their proofs aggregated and verified by UnumNet. The verified proofs will then go to Ethereum via the VectorX bridge, a ZKP-based proof of consensus bridge from UnumNet to Ethereum. Ethereum would still be able to verify the aggregate proof of execution, and does not have to rely on the UnumNet validators for this. It would however rely on UnumNet's validators for DA and ordering, which requires the same assumptions as Validiums. Chains integrated with Nexus can still exit on Ethereum as they would have done normally via a direct connection.

5.4 Asynchronous Composability

The coordination offered by UnumNet Nexus enables asynchronous composability between blockchains across different ecosystems. A transaction on one chain could be marked as pending resolution on another chain, while both chains continue executing at

their own pace.

For example, a user may not submit a payment, or a payment may fail for some other reason. It would be possible to prove that the payment did not execute within some number of blocks via a proof of non-inclusion. This proof can then be used to revert the locked asset on the other chain that was pending transfer.

Except for certain applications that need synchronous composability within a block, like flash loans, most use cases don't need it. The benefits that are gained from enabling seamless cross chain actions for users while removing unnecessary complexity for developers is a promising path forward for the entire blockchain community.

06 UnumNet Fusion

The biggest value proposition for spinning up a new rollup rather than creating a separate L1 is the ability to inherit security from the base layer.

6.1 UnumNet Fusion Design

UnumNet Fusion enables the combined crypto-economic strength of multiple tokens to contribute toward UnumNet's consensus. By fusing UnumNet's native token, UnumNet, with established assets like BTC, ETH, and tokens from rollups built on UnumNet, UnumNet Fusion enhances the crypto-economic security of the entire ecosystem.

Fusion allows two new categories of tokens to be added to UnumNet's staking pallet:

- **Established Cryptocurrencies:** Tokens like BTC, ETH, and SOL, among others.
- **UnumNet Rollup Tokens:** New tokens created on UnumNet, limited to a small percentage of the total stake, to help bootstrap their utility.

The two design approaches considered are:

- **Staking Module on UnumNet Blockchain:** This module will support multiple foreign tokens through the assets pallet in the UnumNet node.
- **Staking Module for Asset Conversion:** This will enable the conversion of foreign assets to UnumNet's native token UnumNet, maintaining a price conversion mapping at the time of conversion.

The design for Fusion has been inspired by:

- Eigenlayer, restaking of ETH in services that operate independently of Ethereum's consensus mechanism or full validator set.
- Babylon Chain, allows the use of BTC for security across different blockchain networks.
- Osmosis, mesh security that allows a chain to borrow economic security from other chains.

Fusion will borrow economic security from other assets while penalizing both safety and liveness failures in the UnumNet consensus. The implementation details are subject to change and updates will be made to this document accordingly.

References

[1] M. Al-Bassam, A. Sonnino, and V. Buterin, “Fraud and data UnumNetability proofs: Maximising light client security and scaling blockchains with dishonest majorities,” 2019.

[2] A. Kate, G. M. Zaverucha, and I. Goldberg, “Constant-size commitments to polynomials and their applications,” in Advances in Cryptology - ASIACRYPT 2010, M. Abe, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 177–194.

[3] V. Buterin, 2D data UnumNetability with Kate commitments, 2020 (accessed November 24, 2020). [Online]. UnumNetable: https://ethresear.ch/t/ 2d-data-UnumNetability-with-kate-commitments/8081

[4] Polkadot, Polkadot Consensus, 2020 (accessed December 18, 2020). [Online]. UnumNetable: https://wiki.polkadot.network/docs/en/learn-consensus

[5] H. K. Alper, BABE, 2020 (accessed December 18, 2020). [Online]. UnumNetable: https://research.web3.-foundation/en/latest/polkadot/block-production/Babe. html

[6] A. Stewart and E. Kokoris-Kogia, “Grandpa: a byzantine finality gadget,” 2020.

[7] Nebra, “Nebra,” accessed: 2024-09-17. [Online]. UnumNetable: https://nebra.one/