Demos: A BORDERLESS INTERCONNECTIVITY LAYER, ENABLING SEAMLESS INTEROPERABILITY AND ADVANCED FUNCTIONS ACROSS ALL CHAINS AND WEBS

Azhar Nurhussen | Cristiano Frassineti | Jacob Ortiz Hansen
team@kynesys.xyz

7th Revision

Abstract

This paper presents a comprehensive analysis of Demos, a novel blockchain system that integrates various advanced concepts and technologies to create a secure, robust, efficient, scalable, and adaptable network. The design incorporates features such as Dynamic Validation Shard Size and Shard Generation with a Common Validator Selection Algorithm (CVSA).

Dynamic Validation Shard Size enables the network to dynamically adjust the number of subdivisions (shards) based on transaction load and node distribution. This ensures optimal performance and resource utilization across varying network conditions. Load balancing techniques distribute the workload and nodes across shards, improving transaction processing speeds and reducing latency.

Shard Generation with CVSA ensures consistent and secure Shard configuration by creating a shared, deterministic source of randomness. The process involves initiating an epoch, applying the CVSA, verifying and distributing the output, generating shards based on the output, and assigning nodes to these shards. This approach enhances security, scalability, and fairness in the network.

The paper also addresses other important aspects, including on-chain Web2 integration and cross-chain logic execution.

On-chain Web2 integration allows native access to Web2 resources within the blockchain network, leveraging cryptographic algorithms for authenticity and integrity verification. Cross-chain logic execution enables scripting, execution, and consolidation of smart contracts and transactions across multiple networks, combining their strengths and mitigating limitations.

Through the integration of these advanced concepts and technologies, the proposed blockchain system design demonstrates the potential for creating a highly secure, scalable, and efficient network suitable for a wide range of applications.

It offers innovative solutions to incentivize node participation, adapt to varying network conditions, ensure secure communication, and provide efficient storage capabilities in a (highly) decentralized manner.

Demos node implementation code can be found at: https://github.com/kynesyslabs/node

Demos SDK code can be found at: https://github/.com/kynesyslabs/sdks

Table of Contents

License

Introduction

The following document presents Demos, an innovative decentralized network that interconnects and expands existing blockchain technology to address the limitations and challenges faced by traditional blockchain systems. Demos introduces novel concepts and mechanisms that aim to revolutionize the way decentralized networks operate and deliver trust, transparency, and scalability.

This yellow paper elucidates the key principles, design, and functionality of Demos, aiming to provide a comprehensive understanding of its inner workings. By exploring the decentralized nature of Demos and its nodal composition, we delve into the intricate mechanics that underpin this groundbreaking network.

To help with understanding the main terms used in this paper, please refer to the Glossary.

Demos Core Structure and Principles

Demos represents a decentralized blockchain system, encompassing a distributed framework consisting of interconnected nodes.

Demos is based on the Demos Network specifications, SDK and framework.

As a **fully self-contained blockchain**, built from the ground up and designed to facilitate the creation of an Omniweb, Demos includes several features and modules.

This paper reflects the current state of the Demos Network specifications as of September 2025.

In the following text, those features will be discussed in detail.

Overall Architecture

As described in the figure below, Demos is able to accept and process different types of transactions coming from and going to **different contexts** through its network of nodes.

A context is defined as the domain in which a data source can be found. For example, to retrieve a resource hosted on a website, the context is Web2.

The two main contexts in which Demos operates are **Web2** and **XM** (Cross-chain Media). For each of them, Demos defines a strict set of rules and workflows to ensure integrity and security.

Furthermore, Demos is able to act at a native level, processing logic and transactions within the context of the Demos L1 itself.

Those rules will be discussed in the following paragraphs.

Nodes and Clients

The term "node" refers to an entity that follows the Demos specifications by providing a client/server architecture that is **compatible with other nodes** of the same network.

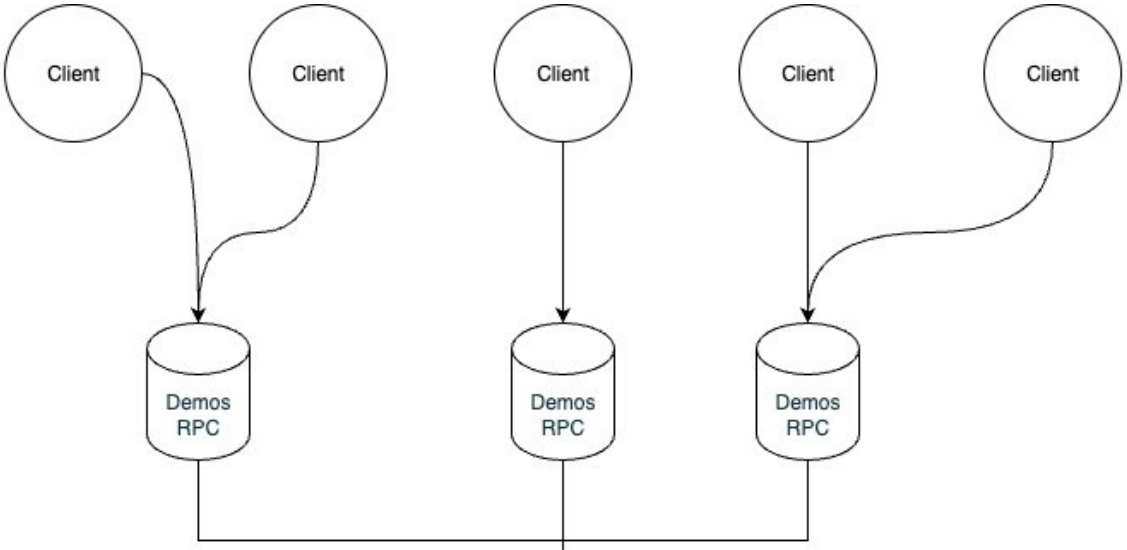
Given the public nature of Demos specifications, it is both feasible and strongly recommended to develop Demos clients capable of actively engaging in the Demos network. This approach fosters **client diversity** and enhances the overall functionality of the network.

Developing a node client is as easy as **implementing the Demos specifications** defined within the official client and exposed at https://kynesyslabs.github.io/demosdk-api-ref/.

Nodes and Validators

The Demos architecture is designed to enforce a possible validator role for every node within the Demos network. As a result, the Demos network achieves a decentralized blockchain structure, leveraging **a particular consensus mechanism** that heavily relies on the resilience and security principles inherent in the blockchain paradigm.

In order to ensure the commitment of validators and mitigate the risk of majority attacks, the operation of a node is contingent upon meeting **specific prerequisites akin to staking**. This approach aims to incentivize and reward honest and productive participants within the network while penalizing those who engage in malicious activities or exhibit inadequate performance.



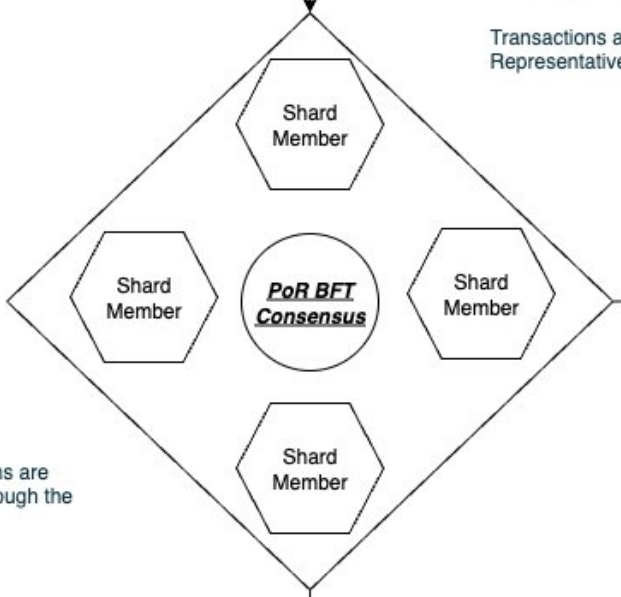
Payload Types

Each Demos Transaction could be composed by any payload supported by Demos: Web2, XM or mixed payloads are all encapsulated in a Demos Transaction object.

DTR Propagation

Propagation

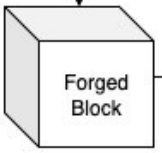
Transactions are validated and sent to the Representative Shard for mempool inclusion.



*This is known as the PoR Shard and is calculated using the **CVSA***

Execution

During Consensus Time, Transactions are executed by the Shard Members through the appropriate Demos module.



Notably, the block contains and sets all the necessary properties to calculate the next block's CVSA and thus individuating the next PoR Shard

Synchronization

Forged blocks are synced by the syncing mechanism to the RPCs connected to the Representative Shard.

The Role of Demos

In order to grasp the essence of Demos' purpose and its potential capabilities, it is imperative to gain a comprehensive understanding of the network's fundamental role, delving into the intricacies of its multifaceted offerings.

Demos, functioning as a network-agnostic **Omniweb** infrastructure, has been meticulously architected to **facilitate seamless communication**, interaction, and mediation among the realms of Web2, Web3, and the myriad of existing blockchains.

While Demos acknowledges and **recognizes the inherent value and benefits offered by the diverse L1, L2, and L3 blockchains currently in existence**, its primary objective is not to undermine their utility. Rather, Demos assumes the role of **decentralizing the blockchains themselves**, empowering users and developers to harness the optimal attributes embedded within each distinct chain.

As explained in this paper, all the features that Demos provides are made with the specific goal of **enhancing decentralization without reinventing the wheel**. Each technology and blockchain possesses a unique set of strengths and weaknesses, that if combined would enhance the entirety of Web3.

It is within this context that Demos emerges as a solution, a **data layer** that enables seamless interoperability among different technologies. Moreover, Demos enriches the entire ecosystem by incorporating its own distinctive features, thereby fostering a cohesive and unified environment.

A Trustless Data Layer: towards the Omniweb

While being suitable for a vast array of operations, the main Demos goal is to establish itself as a Data Layer, or **Data Facilitator**. The core philosophy of Demos is that there are great projects and working infrastructures that suffer from **being isolated from each other**, with the best case being a sturdy, non standard compatibility.

We call all of these isolated systems “**contexts**”. If we see each one of these contexts (such as blockchains, Web2 protocols and so on) as railways, we see a huge number of working railways whose trains (data) can’t travel on the others. Demos is a **meta-railway**: the railway that connects and makes compatible other contexts using a common interface.

An intent-based environment

The Demos network is designed to be **intent-centric**. While the term can be confusing as it applies to many possible mechanisms, it is very easy to explain how the Demos blockchain manages intents.

Each transaction contains information that **assigns an intent** to a client with **no degree of uncertainty**. This means that the client expresses its will to perform the operations contained in the transaction at a given timestamp, with a given target block and in a reproducible manner.

Thanks to this, Demos Network is able to **assign a precise chronological order** to its shared mempool: **frontrunners and priority fees are not supported** in this scenario. Once the transaction has been correctly executed by the blockchain, finality can be guaranteed **even before the next block is mined**.

In this mechanism, the inclusion of a transaction in a block is an **acknowledgement of the integrity** of the data and the retrieval process that led to the execution of the included payloads.

Each transaction is essentially an **intent** that, once resolved by the network through the mechanisms explained in this paper, is binding and already executed. It should be noted that Demos Network acts as a **supervisor** of the execution of intents: for this specific reason, it guarantees impartiality and certainty even within the same block.

Transaction Lifecycle: Scaling through DTR

The Demos network is engineered to process a high volume of transactions per second.

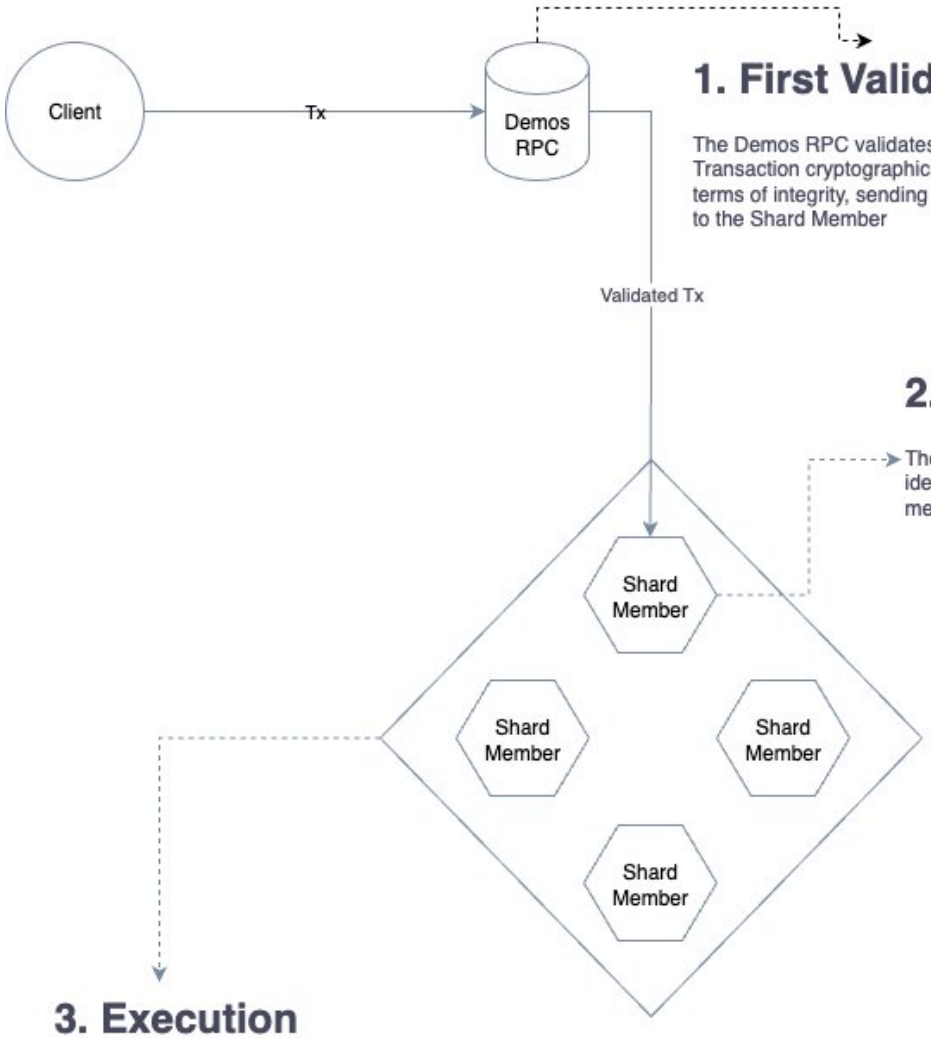
Consequently, each node (also referred to as an RPC endpoint) must efficiently handle substantial loads. To achieve this, the Demos architecture implements a mechanism known as 'Decentralized Transaction Relay' (DTR).

Every transaction adheres to a **precise lifecycle** designed to ensure execution, security, and integrity from inception to finality. This lifecycle leverages the Proof of Representation (**PoR**) component of the consensus mechanism and its associated **CVSA** feature. The process unfolds as follows:

1. A transaction is received by an RPC endpoint, which **verifies** the transaction's signature and hash, alongside performing basic integrity checks.
2. Upon successful completion of these initial checks, the transaction is **relayed** to a randomly selected Shard member designated for the upcoming consensus round.
3. **Leveraging the CVSA feature**, each RPC endpoint is aware of the nodes constituting the Shard once the preceding block has been forged.
4. The designated Shard member stores the transaction in its **local mempool** and concurrently disseminates a copy to another Shard member to enhance availability and redundancy.
5. The transaction is subsequently organized within the **PoR Shard's shared mempool** upon commencement of the consensus cycle.
6. Once the entire Shard reaches agreement on the shared mempool's state, the transaction is executed, and the **Global Change Registry (GCR) is updated**.

This architecture enables RPC endpoints to **efficiently manage a high volume of transactions**, as Shard members can readily store them pending the consensus phase.

Such a decentralized approach to transaction processing ensures network-wide integrity, order, and finality, while minimizing overhead associated with execution, verification, and potential rollbacks of invalid GCR updates.



1. First Validation

The Demos RPC validates the Transaction cryptographically and in terms of integrity, sending the valid ones to the Shard Member

2. Mempool Inclusion

The Shard Member verifies the Demos RPC identity and insert the Transaction into its own mempool

3. Execution

During the Consensus Loop, the PoR Shard merges, validate and execute Transactions in order thus modifying the GCR and forging a new block

Native Bridges: Ensuring Secure Cross-Chain Value Transfers in Demos

To operate efficiently across diverse networks, Demos incorporates **built-in bridging functionality**. This dynamic feature enables Demos to assess, transfer, and manage liquidity across different chains by leveraging the decentralized architecture of the entire network.

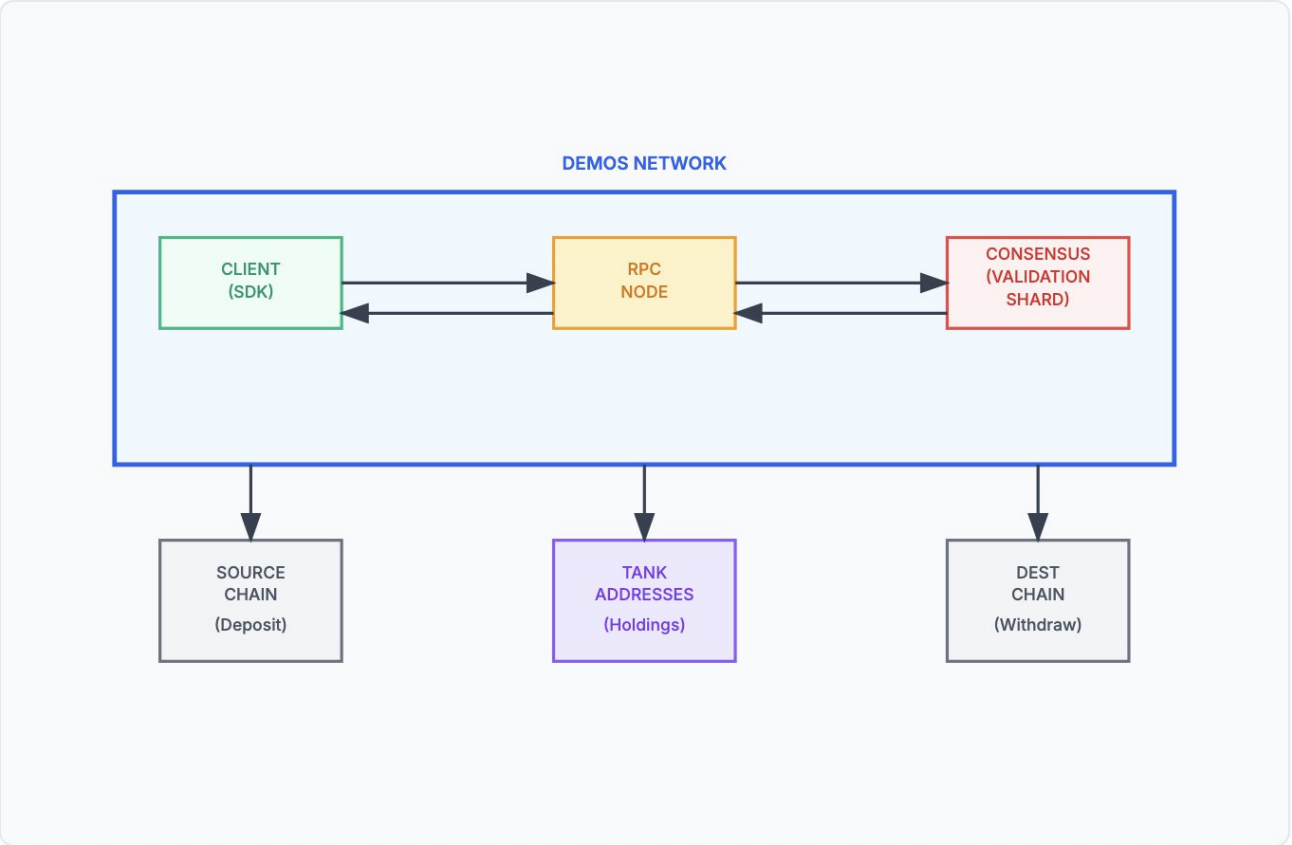
Each Demos RPC endpoint accepts a specific operation type, identified by the **nativeBridge** field. Such an operation signifies **the client's intent to execute a liquidity transfer** between two different blockchain contexts (e.g., Solana and Ethereum).

Upon receipt, the nativeBridge operation is processed by the RPC endpoint, which infers the relevant Liquidity Tank for the source network and the settlement currency (typically a stablecoin like USDC).

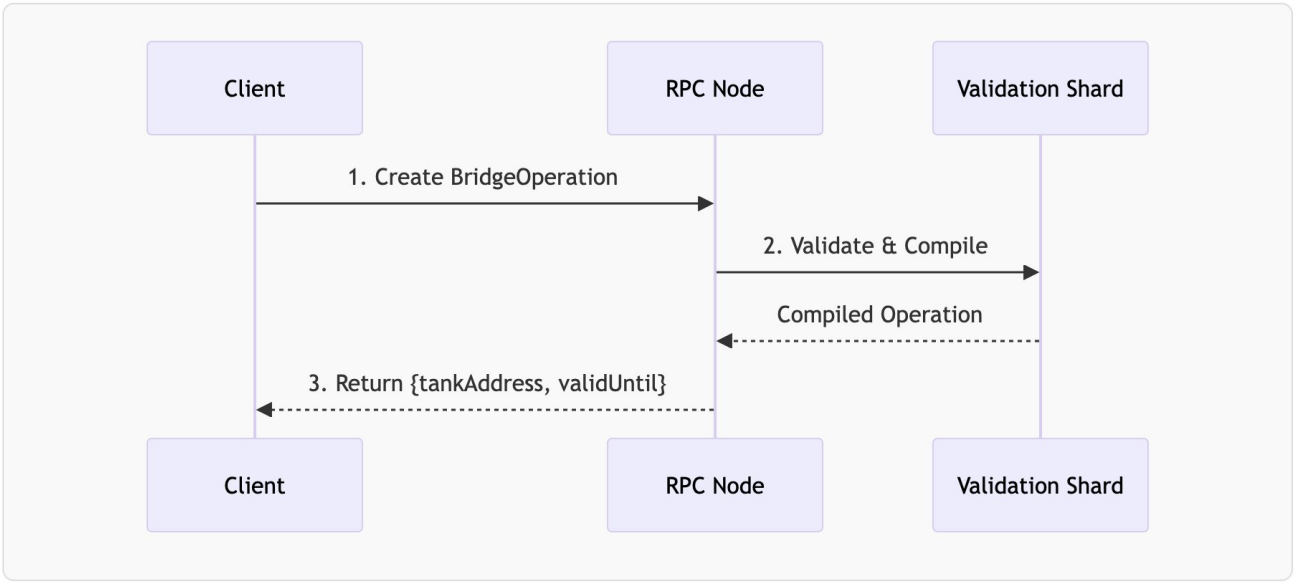
The processed operation details are then returned to the client. The client, in turn, **generates a transaction incorporating this validated operation.** This transaction is handled like any other within the Demos network and is subsequently **submitted to a Shard member** for execution during the consensus phase (see 'Transaction Lifecycle').

During the consensus process, a nativeBridge transaction undergoes checks for validity, compliance, and **fulfillment** of execution conditions. If the client's initiating transfer (deposit) on the source chain is confirmed and all checks pass, the funds are transferred from the Demos Liquidity Tanks to the recipient's address on the destination chain.

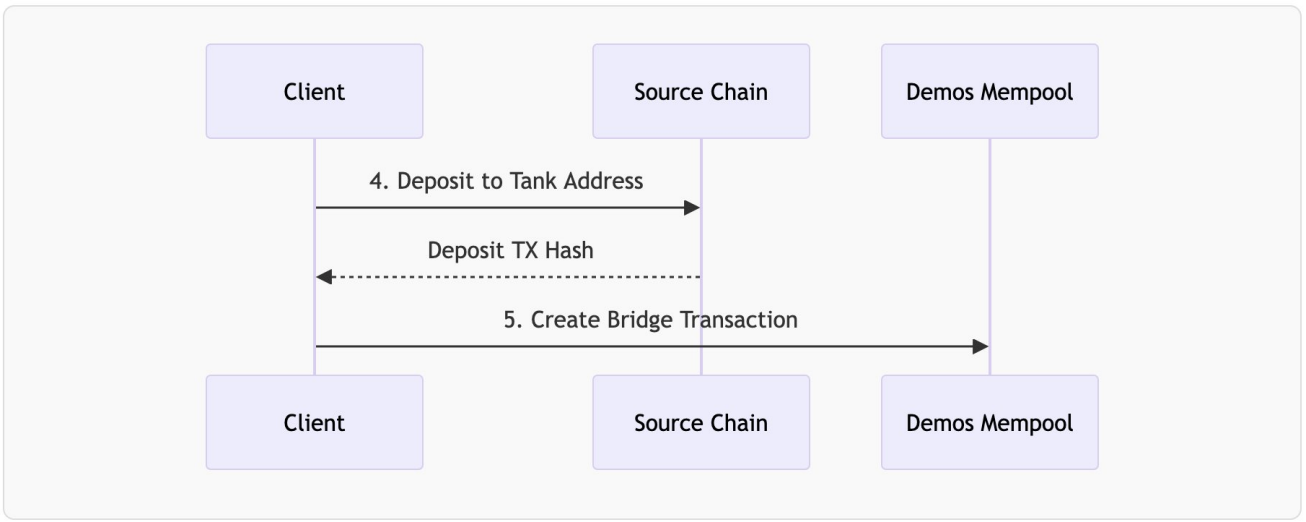
To maintain security, the state of authorization of each Liquidity Tank is **refreshed every block.** Furthermore, releasing funds from a Tank requires signatures from a majority of the respective Shard members.



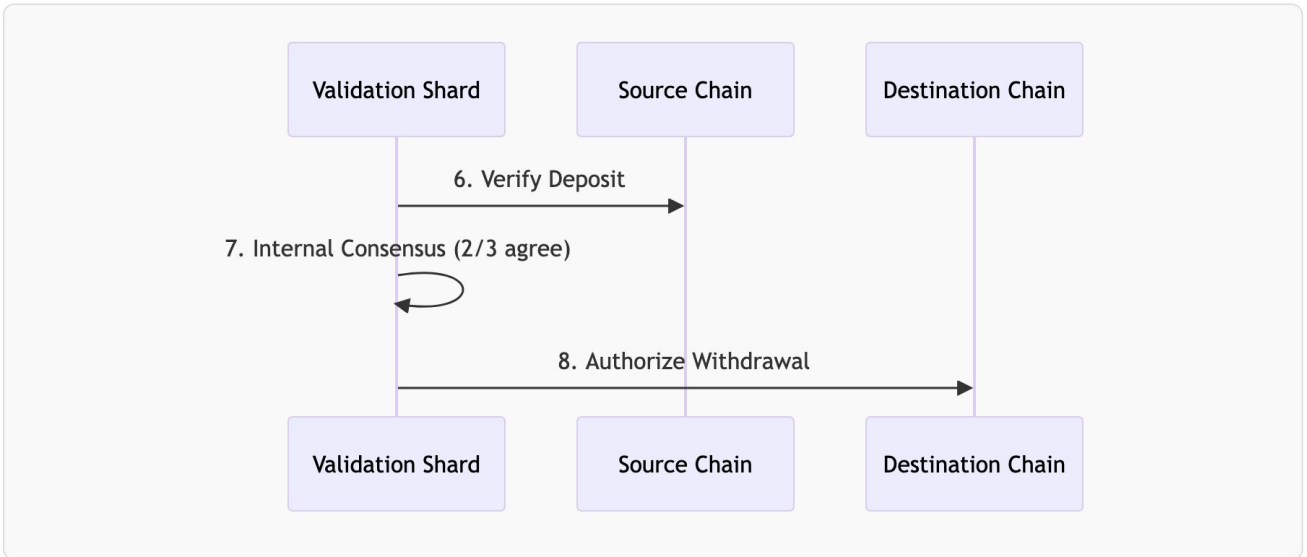
Phase 1: Client Request & Validation



Phase 2: Deposit Execution



Phase 3: Consensus & Withdrawal



Liquidity Tanks: Demos Metafees State and reserve

Being isolated and self-sufficient, **each blockchain requires fees in its own currency** (with some exceptions). What's more, it's very likely that clients won't have every single native coin for the chains they want to operate on when operating through an interconnected layer such as Demos.

If a client wants to complete an XM transaction with writes on, for example, chain A, B and C, **they would normally have to provide A tokens, B tokens and C tokens for gas, plus the Demos network fees.**

This option is **inconvenient** and unacceptable. In order to harmonize gas requirements, Demos creates entities called Liquidity Tanks on different networks, powered by the above described Native Bridges. Depending on the network architecture, a Liquidity Tank can be a smart contract, a wallet, a chainware or any other compatible form.

One of the roles of the Liquidity Tank is to **always be in real-time contact with the Demos Network** and to collect fees in the chain's native currency or (if supported) in the chain's specific token. The Demos Network **allows developers to specify how they wish to pay for gas**, so that a developer can quickly execute their transaction with the tokens they hold.

In the background, each gas tank is known to the network and provides its capacity so that the Demos Network always knows the total gas balance and how it is distributed.

An example might be as follows:

Alice sends an XM Transaction involving a write operation on a EVM network and two L1s

Demos Network calculates the max fees requested in the various steps and presents Alice with coherent options to pay gas

Alice, holding just the EVM network native token, pays with that network token

The EVM Network gas tank is filled and synced, while the RPC converts (if needed, usually done in advance) the needed tokens

This way, Alice **just has to hold a single token instead of 3.**

Demos Architecture & Consensus Mechanism

Decentralized Adaptive Architecture (DAA)

Overview

Decentralized Adaptive Architecture (DAA) is the name of Demos architecture model, designed to achieve exceptional efficiency, scalability, and performance in decentralized systems while operating as a meta-chain infrastructure. DAA leverages dynamic system adaptation and representative sharding technology to ensure robust security.

Mechanism

1. Global Change Registry (GCR)

To enhance scalability and efficiency, DAA implements the Global Change Registry (GCR) paradigm. This mechanism involves processing transactions in the most decentralized and collaborative way so that it is easy, fast, and efficient to find and scrap invalid or malicious payloads. Moreover, the Global Change Registry ensures consistency across shards while allowing them to operate independently.

2. Proof-of-Representation with Byzantine Fault Tolerance (PoR-BFT)

DAA employs a two-layer byzantine consensus mechanism, combining Proof-of-Representation (PoR) with Byzantine Fault Tolerance (BFT) for processing the Global Change Registry. Nodes are selected to form a **Representative Shard** for the validation process based on the **PoR algorithm** that ensures a deterministic non-tamperable selection method. This incentivizes stability and efficiency. Within the Representative Shard, a **Byzantine Fault Tolerant** consensus protocol is implemented, guaranteeing robustness, security, and scalability without burdening the entire chain.

3. Dynamic Validation Shard Size

DAA continuously adapts to varying transaction volumes by **dynamically adjusting the number of participants** to the block proposing consensus phase. This adaptability is achieved through a randomness layer powered by a **Common Validator Selection Algorithm**, ensuring the generation of a dynamic and consistent Representative Shard.

Advantages

1. **Enhanced Scalability**: Through the implementation of the Common Validator Selection Algorithm, the Decentralized Adaptive Architecture (DAA) system achieves superior processing capabilities by concurrently handling a multitude of transactions within a deterministic representation of all the nodes. This substantially augments the network's overall capacity, enabling seamless scaling.

2. **Robust Security**: The DAA system incorporates the classic Byzantine Fault Tolerant (BFT) consensus technique in conjunction with the Proof-of-Representation model to establish a formidable security framework that effectively mitigates a wide range of potential attack vectors. This amalgamation of methodologies ensures the integrity and resilience of the system against malicious activities.

3. **Heightened Efficiency**: DAA demonstrates remarkable efficiency by dynamically adjusting the proposers' count in response to the network's load. Leveraging the power of BFT, independent cross-node validation is facilitated, leading to optimized processing speeds and resource utilization. The system operates at peak efficiency, bolstering overall performance.

4. **Sustained Decentralization**: By employing a democratic, secure, and tamper-proof process for validation Shard generation, integrated with the Proof of Representation (PoR) consensus component, DAA effectively addresses performance problems arising from communication overhead and local transaction processing. Shard members are determined at the commencement of each block cycle, with their composition mathematically proven to be consistent across all legitimate nodes.

The Global Change Registry (GCR)

GCR is a mechanism that allows transactions to be declared with the possibility of rolling back, examining, and verifying their content even before a block is emitted.

Mechanism

1. A transaction reaches an RPC and is validated for identity and integrity. The signature and the hashes are examined and confirmed, **then the transaction is sent to a Shard member**.

2. During the consensus stage, Shard members **merge their respective local mempools** to construct a unified, ordered mempool. This consolidation yields a mempool encompassing all unique transactions processed by RPC endpoints during the current block period. Following verification, these transactions are then dispatched to the **GCR Preprocessor**.

3. Global Change Registry (GCR) Preprocessor: The GCR functions as a comprehensive ledger that maintains the **state of the entire blockchain network**. The Preprocessor examines the transaction and builds a delta (a sorted registry of changes) to the Global Change Registry without modifying it directly.

4. Each subsequent transaction follows the same process as above. Once a block needs to be forged from the shard unified mempool, all of the GCR Preprocessor registry items are **executed one by one** in the right order on a copy of the GCR.

5. If a transaction produces an invalid result in the GCR, it is rejected or reverted. Otherwise, **the GCR gets permanently modified once the block is emitted by the consensus mechanism**.

Key Features and Solutions

1. Predictive Transaction Validation

Thanks to the local GCR, each node is able to predict with reasonable precision if a transaction will fail or will succeed. Even before the next block is emitted, many elements and mechanisms allow a transaction to be **simulated with efficiency and rapidity**.

2. Resource-Friendly Rollbacks

For the same reason, as **the GCR is not enforced until the block is emitted**, it is very easy to track and roll back invalid, malicious, or malformed transactions. Even if a transaction makes it into the GCR registry, once the consensus validator parses it and flags it as invalid **there is no need to do anything else other than scrapping the transaction**.

Advantages

1. Scalability

GCR leverages **parallel processing of transactions** within individual nodes and employs efficient global state management techniques to significantly enhance the network's capacity.

2. Efficiency

The employed strategy effectively **minimizes cross-node communication overhead** and optimizes GCR updates, thereby achieving a notably high level of operational efficiency.

3. Consistency

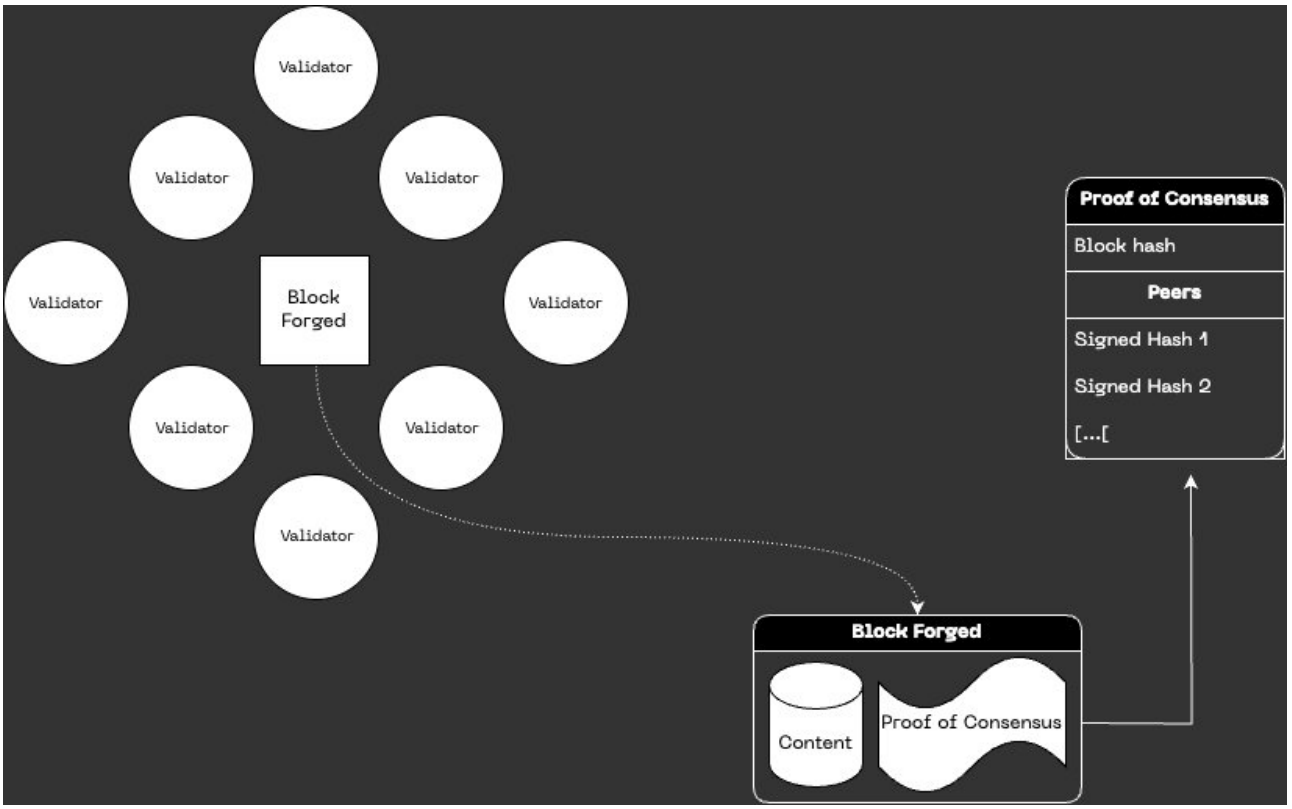
By diligently maintaining a **synchronized state** of the blockchain across all nodes, GCR ensures a **unanimous** agreement among all network nodes regarding the current network state.

4. Enhanced Interoperability

The incorporation of GCR in the system facilitates improved **interoperability among different nodes** within the blockchain network. Consequently, any node gains the ability to seamlessly access the state of any address present across the entire network.

Proof-of-Representation with Byzantine Fault Tolerance (PoR-BFT)

Figure 4: Proof of Representation



Proof-of-Representation with Byzantine Fault Tolerance (PoR-BFT) is an advanced consensus mechanism that operates at a heightened level of technical safety, effectively **ensuring network stability, efficiency, and security** within the Demos Network.

It accomplishes this through the integration of a Byzantine Fault Tolerance (BFT) consensus mechanism within a **Shard representing the entire validator network**, selected with the utilization of the CVSA.

This Shard is called the **Representative Shard** and is composed of the result of an algorithm (the CVSA) that, **using the deterministic property of the transactions** in the GCR, ensures the same group of validators are chosen by all the legitimate nodes without the need for data transfer.

The Two Consensus Tiers

1. Byzantine Fault Tolerance (BFT) within a Shard

The PoR-BFT system leverages the Byzantine Fault Tolerance consensus mechanism. **BFT enables rapid validation and agreement of transactions within the shard**, even when confronted with the presence of malicious or faulty nodes. By mandating the agreement of a supermajority (typically **2/3+1**) of nodes prior to appending a new block to the shard's blockchain, BFT ensures efficient operation and robust security within a group of validators.

2. Proof-of-Representation (PoR) for a Representative Shard

To elect the Representative Shard, the PoR-BFT system employs the CVSA mechanism. Validators selected for this process are chosen based on **deterministic values contained in the GCR** and in the intrinsic blockchain properties, which will always be identical for all the nodes that act on the same, untampered and verified data. This way, **malicious nodes or malfunctioning nodes are automatically excluded** from the consensus selection without any network overhead.

The exact mechanism that allows this to happen is described in the ‘Shard Generation and Validators Choice with a Common Validator Selection Algorithm (CVSA)’ section.

Advantages

1. Efficiency

The utilization of a Byzantine Fault Tolerant (BFT) consensus mechanism within a Representative Shard facilitates **rapid transaction processing** and streamlined intra-Shard operations.

2. Augmented Security

The Proof of Representation (PoR) consensus mechanism significantly strengthens security measures during the validators' election process for the next consensus round. By selecting nodes using a deterministic approach based on data integrity, PoR effectively **mitigates the risks associated with malicious activities**, fostering heightened integrity and efficiency within the blockchain system.

3. **Scalability**

   The PoR-BFT approach offers an advanced and **scalable** solution that can effectively manage a substantial volume of transactions. This approach ensures consistency and reliability within the GCR, even in the face of high transaction loads, thereby enabling the system to seamlessly handle increased demands without compromising performance. This is done by changing the **size of the Representative Shard** to facilitate computation and data transfer even in the case of massive traffic.

4. **Stability**

   The PoR mechanism promotes **stability** within the blockchain network by **avoiding data overhead** and by incentivizing long-term participation from network nodes. By dynamically selecting a Representative Shard for the whole blockchain, PoR fosters an environment of reliability and predictability, ensuring the overall stability of the blockchain system.

Shard Generation and Validators Choice with a Common Validator Selection Algorithm (CVSA)

The process of Shard generation using a Common Validator Selection Algorithm (CVSA) involves creating a **shared, deterministic source of pseudorandomness** to ensure all nodes generate the same Shard configuration given the same network state.

Phase 1 - Initiate CVSA Prerequisites

At the commencement of each validation round, the blockchain system initiates the CVSA process. This process commences by selecting an initial input, commonly denoted as the "**seed**." This seed is **derived from the last block hash**, which in turn incorporates the Global Change Registry (**GCR**) hash (refer to the GCR section for further details).

Phase 2 - Apply CVSA

The selected seed is then input into a pseudo-random number generator to generate a **CVSA**. The CVSA will be **identical for those nodes compliant with the blockchain** mechanism whilst will be **different for malicious or malfunctioning nodes**. The output remains impervious to prediction until the computation is finalized and needs to be demonstrated by **cryptography**, ensuring security and unpredictability.

Phase 3 - Verification and Distribution

Once the CVSA has generated the output, all other nodes will possess **the same information without the need for transmission.** The inherent property of rapid verification in the CVSA guarantees simultaneous propagation of the output to all nodes.

Phase 4 - Shard Generation

Each node in the network uses this output as the **seed for Shard generation** and validation assignment. Since the seed is identical for all nodes, the Shard generation process results in the **same set of shards across the entire network**. The nodes are then assigned to these shards based on the deterministic process using the same seed, ensuring uniform distribution and a balanced workload.

Phase 5 - Begin Epoch Operations

Following shard generation and node assignment, the blockchain network proceeds with its operations for the new epoch, **processing transactions within the Shard once the consensus cycle begins**, and committing state changes to the Global Change Registry (GCR) in a predictable yet secure manner.

The utilization of a CVSA within the randomness layer provides a secure, verifiable source of randomness, enabling consistent Shard generation across the network, enhancing security, and ensuring scalability.

Dynamic Validation Shard Size

Dynamic Validation Shard Size entails the implementation of a sophisticated approach wherein the number of nodes chosen within the network to operate the consensus, known as Representative Shard, can be **dynamically adjusted** in response to variations in transaction load and node distribution. This adaptability grants the blockchain network the ability to uphold optimal performance levels across a diverse range of conditions, spanning from scenarios of low to high transactional demand.

Design and Mechanism

1. Choosing a Representative Shard: each time a consensus cycle is completed by the validation shard, the Demos blockchain utilizes a deterministic algorithm that returns the same exact result on all the nodes that have the same data in their mempool and in their blockchain. This way, **all the nodes are able to instantly know who the participants will be in the next Shard** without any communication needed.

2. Parameters: to create the Representative Shard, parameters based on the chain **load, throughput, and specific conditions** are enforced and added to the above mentioned initial seed to determine the size of the Representative Shard.

Recap

This newly proposed blockchain consensus mechanism presents a robust, scalable, and innovative solution, integrating features of Global Change Registry (GCR), Proof-of-Representation with Byzantine Fault Tolerance (PoR-BFT), Dynamic Validation Shard Size, and a secure Randomness Layer powered by a Common Validator Selection Algorithm (CVSA).

Through GCR, the system **efficiently manages transaction processing** and data storage across various shards, enhancing scalability while ensuring a consistent view of the global state.

The PoR-BFT consensus model integrates a Byzantine Fault Tolerance consensus within each Shard with a Proof-of-Representation mechanism for global block proposals, optimizing both **intra-Shard and inter-nodes operations.**

Dynamic Validation Shard Size enhances the system's **adaptability**, ensuring optimal performance and resource utilization regardless of network load. This, combined with the integration of a CVSA-powered Randomness Layer, ensures secure, verifiable, and **consistent** Shard generation, further contributing to the network's robustness and security.

CVSA provides an innovative approach to generating shared randomness that enables the system to maintain **uniformity and fairness** in the node distribution and workload allocation, further enhancing the network's scalability.

The Demos blockchain architecture design showcases the potential of integrating various advanced concepts and technologies in building a blockchain network that is not only secure, robust, and efficient but also highly scalable and adaptable to varying network conditions.

The Importance of Being Futureproof: Post Quantum Cryptography

Blockchains depend heavily on cryptography to ensure a robust level of security while handling highly sensitive information. For an algorithm to be considered secure, it must stand the test of time, a criterion met by widely used algorithms such as ed25519 and RSA.

These algorithms are particularly favored for their implementation in user-friendly environments. However, the ability to withstand the test of time is not infinite for ed25519 and RSA, and this limitation applies to blockchains that utilize different, often weaker, algorithms as well.

The emerging field of quantum computing poses a significant challenge. Quantum processors, while not yet consumer-grade, have already shown substantial improvements in efficiency and computational power. The advent of quantum computing becoming accessible to the general public, or at least to a well-connected subset, is not far off.

So, what is the solution if even the most secure algorithms may eventually fail? The National Institute of Standards and Technology (NIST) is actively encouraging the development of quantum-resistant cryptography. They regularly publish a list of recommended algorithms for public study.

Demos, designed to withstand this issue, is capable of switching its cryptography and hashing modules to entirely new algorithms upon consensus. Post Quantum Cryptography modules are already integrated into the Demos source code, awaiting activation or updates. While we are actively testing and verifying highly regarded algorithms such as Lattice-based solutions, the Demos network is designed to continuously update its "backup" algorithms.

Demos Network specifies and ships additional cryptography modules exposing the exact same interfaces as the default modules.

Demos supports both ed25519 and RSA for identity management and encryption. Those algorithms are more than battle proven but they are not guaranteed against Quantum attacks.

That’s why Demos specifications are constantly updated to reflect the cutting edge algorithms available for implementation. If, in any moment, the Network should be in need of using those algorithms, through the usual consensus they can be used as drop-in replacements without any downtime or compatibility issue.

Demos currently implements a system named Unified Cryptography, which provides easy-to-use APIs for developers and users, enabling the seamless utilization of various cryptographic schemes. Leveraging clear specifications, Demos nodes can determine the appropriate cryptographic scheme based on the client's requirements (whether a dApp or a developer).

The Post-Quantum Cryptography (PQC) Module is perhaps the best example of how modularity can benefit a blockchain in such a context, whilst also allowing us to delve deeper into this aspect of blockchain development.

The Demos PQC Stack

The Demos Network specifications, inherently dynamic due to ongoing research in the Post-Quantum Cryptography (PQC) field, present a selection of algorithms balancing efficiency and security. We support state-of-the-art algorithms by closely adhering to NIST recommendations.

The chosen algorithms are:
– ML-DSA
– Falcon
– ML-KEM with AES

Each one serves a specific purpose.

ML-DSA

The ml-dsa, a digital signature scheme, is the module-lattice (ml) based algorithm known as Dilithium. Demos implements the ml-dsa-65 variant, which includes the following features:

Public Key Size (Bytes)	Private Key Size (Bytes)	Signature Size (Bytes)
1952	4032	3309

This schema targets level 3 of NIST recommendations.

You can read more about ml-dsa at https://openquantumsafe.org/liboqs/algorithms/sig/ml-dsa.html

Falcon

Falcon, another digital signature scheme, is distinguished by its smaller key and signature sizes relative to ml-dsa. Although it offers a marginally reduced level of post-quantum security, it is still deemed adequate. Demos implements Falcon-512, incorporating the following features:

Public Key Size (Bytes)	Private Key Size (Bytes)	Signature Size (Bytes)
897	1281	752

This schema targets level 1 of NIST recommendations.

You can read more about Falcon at https://openquantumsafe.org/liboqs/algorithms/sig/falcon.html

ML-KEM with AES

The ml-kem is a Key Encapsulation Mechanism (KEM), a cryptographic scheme designed for secure key establishment. It provides a quantum-resistant method where a shared secret is securely encapsulated for a specific recipient using public-key cryptography. This enables the communicating parties to establish a symmetric key for subsequent secure communications. Demos implements ml-kem-768 with the following features:

Public Key	Private Key	Ciphertext	Shared Secret	Keypair Seed
1184	2400	1088	32	64

This schema targets level 3 of NIST recommendations.

You can read more about ml-kem at https://openquantumsafe.org/liboqs/algorithms/kem/ml-kem.html

To ensure data confidentiality, secrets established via ml-kem are utilized with the well-established AES algorithm for symmetric encryption.

Demos Modularity

As a highly dynamic blockchain with a growing number of integrations, features and contexts, the Demos network has been designed with modularity in mind since day 0.

Modularity is a broad concept and is often used as a synonym for "extensibility". While modules can certainly extend the capabilities of a network, it wouldn't be fair to exclude the other key benefit: maintainability.

In the following paragraphs we will examine both, as they are defined in the Demos network specifications.

Extensibility

The first consequence of modularity is of course extensibility. This means that every Demos client has to follow common and standard practices defined by the Network so that it can quickly update its features if requested.

The Demos Network is able to, through consensus, insert or remove certain features. In that case, the consensus outcome must be followed so that the new features are both implemented and tested.

Maintainability

The second but not less important consequence of modularity is maintainability. As stated above, the Demos Network is able to quickly propagate updates in a safe way, leveraging the blockchain’s consensus and traceability so that both patches and rollbacks are possible without hindering the network security.

On-chain Web2

The establishment of a seamless linkage between Web2 and Web3 serves as a crucial solution to a significant impediment hindering the emergence and flourishing of many projects and technologies.

Demos Network leverages its decentralized structure and battle-proven cryptographic algorithms to provide seamless, native, and trustless access to Web2 resources from within blockchains.

Design and Mechanism

At the core of Web2 integration, there is a specific endpoint available to dApps and clients that allows actors to ask for a specific resource such as an API, a GraphQL request or a plain HTTP(S) session. Once the RPC receives a correctly compiled request on the Web2 endpoint, it first validates the request by checking the hash of the request and its signature to ensure authenticity and integrity.

It then creates an HTTP(S) proxy between the client and the requested resource, allowing direct communication between the two, while being able to monitor the communication process to ensure that no tampering or malicious actions have taken place.

While the communication is in progress, the RPC can attest to its validity, integrity and legitimacy without having to decrypt the data it contains. In this way, even if a request is encrypted using HTTPS, the RPC is able to maintain the security of the communication channel without compromising the user's privacy.

The employed hashing algorithm used to guarantee the cryptographic security of the communication channel is SHA256, and the signature method utilized is ed25519. This empowers the on-chain retrieval and provision of Web2 data without reliance on external oracles or centralized solutions.

While this mechanism is secure enough to be trusted, control mechanisms are in place to also ensure performance, error handling, and the resolution of specific exceptions that may occur during the integration of Web2 and Web3.

Securing Web2 Context: Data Agnostic HTTP(S) Relay Proxies

To ensure a verifiable, robust and trustless way of accessing Web2 data on demand, the Demos network follows strict specifications to be able to build, spawn and control a chain of trust.

This means that a Web2 transaction follows a very rigorous workflow that is designed to avoid both data tampering and uncertainty.

To better illustrate the above explanation, let's examine the typical flow of a Web2 transaction:

1. The client produces a valid Web2 transaction and sends it to the chosen Demos RPC

2. The Demos RPC generates a dedicated DAHR **(Data Agnostic HTTP(S) Relay)**
   a. Proxy

3. The DAHR proxy is presented to the client, which uses it to communicate natively with the requested resource.

4. The client is now able to process and maintain proper communication with the Web2 resource without the classic limitations of an on-chain oracle.

5. At the same time, the RPCs monitor this communication channel by observing and certifying it without interfering.

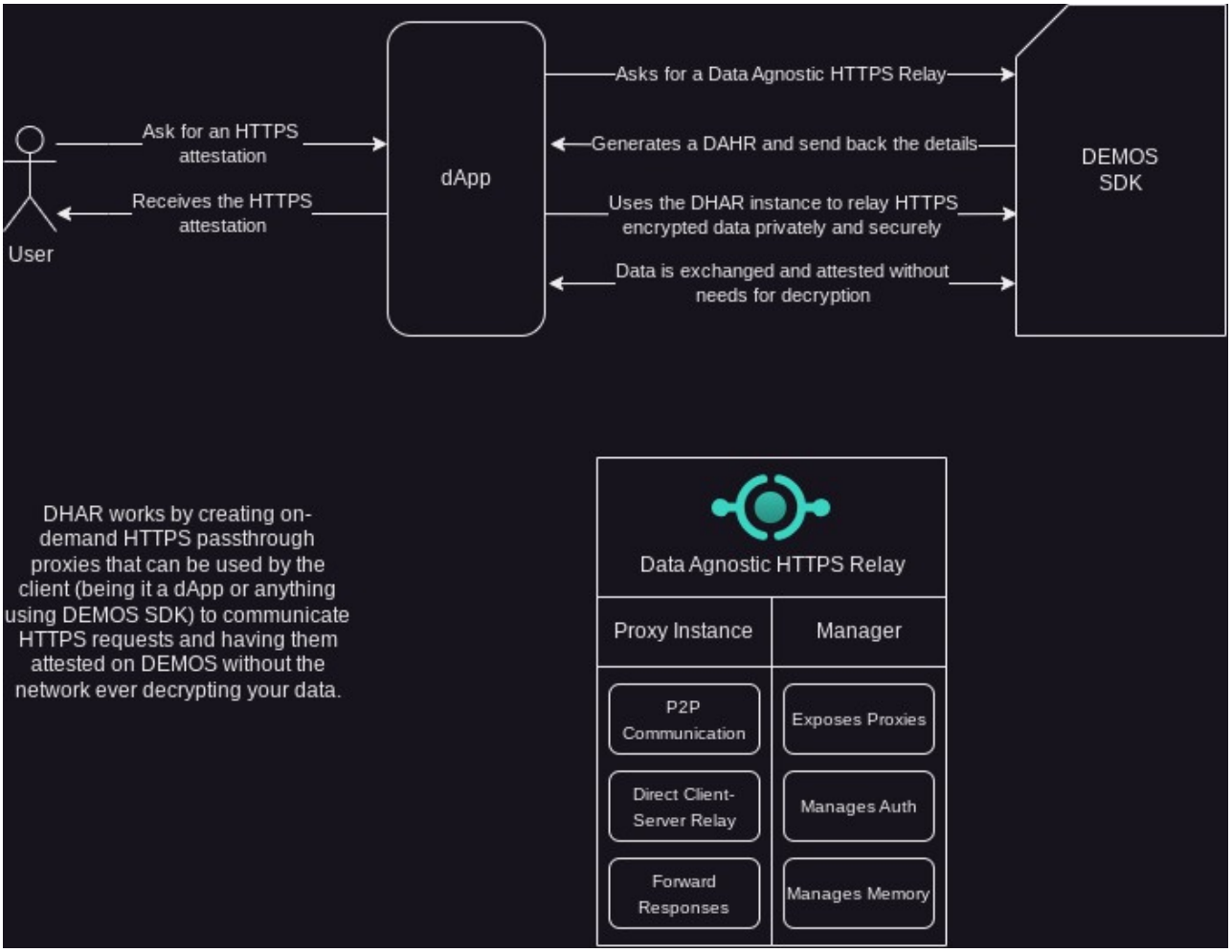
6. In this way, the client not only receives the resource immediately, guaranteeing immediate finality, but is also always aware of the integrity of the communication.

7. When the next block is mined, the current attestation data and status for that DAHR proxy will be added to the blockchain.

This way, while allowing customizations in terms of speed and security, Demos is able to guarantee **integrity and verifiability of the data provided**.

The novel approach to Web2 data pulling on chain exemplifies the need of a paradigm shift from a push-oriented, controlled and limited offering of oracles to the pull-oriented, on demand and **self-verifiable offering** of data feeds.

Figure 5: Architecture of a DAHR Proxy



Ensuring Integrity and Stability

To mitigate the unpredictable nature of Web2 data retrieval and prevent potential clogging and security concerns, a robust system must be put in place. This system employs a multitude of techniques such as request throttling, End-To-End validation, and replication verification to ensure optimal performance and security.

Request Throttling

To prevent overloading low-performance endpoints and to ensure that responses are received within an acceptable time frame, request throttling is employed. This technique involves limiting the number of requests that can be made within a specific timeframe. By implementing request throttling, Web2 systems effectively manage their resources, preventing congestion and optimizing performance.

Request throttling is implemented using various techniques built on two main paradigms: rate limiters, which restrict the number of requests based on a predetermined rate, and timeframe restriction, which restricts the timeframe in which a request can be retrieved, attested, and sent back.

End-to-End Validation

End-to-End (E2E) validation constitutes a crucial technique for guaranteeing the authenticity of Web2 data retrieval. This process involves thorough verification of the identities of all parts involved before any data exchange takes place. By enforcing E2E validation, Web2 systems effectively counter potential security threats, such as man-in-the-middle attacks and unauthorized access.

On the Web3 side of the request, E2E validation is implemented using on-chain signature-based identity verification, while digital certificates are employed on the Web2 side to ensure the integrity of the request and response flow.

Session Based Approach

Derived from the DAHR Proxies design, Web2 Retrieval on the Demos chain supports and encourages a session-based approach. This means that a client is able to negotiate a fixed, prepaid gas charge with the RPC so that it is able to access data in a continuous way, for example by creating a streaming session with the resource (such as WebSockets), without having to pay a fee each time the resource is refreshed.

Not only does this help to keep costs at an acceptable level for both the client and the RPC, but it also drastically reduces the time required to retrieve a resource as the gas transactions are not necessarily made each time.

Cross-chain Logic Execution

The concept of Cross-chain Execution is the **integration of diverse contexts**, combining attributes like security and speed by empowering developers to script, execute, and consolidate smart contracts and transactions **across multiple networks**.

By harnessing the **strengths of individual networks** and mitigating their respective limitations, Cross-chain Execution establishes a more efficient and secure system as a whole. For instance, developers can **leverage** the speed and efficiency of one network while **compensating** for its lack of robust security features by utilizing another network that offers enhanced security capabilities.

This approach offers flexibility and adaptability, enabling developers to select the most suitable network for each specific task, unrestricted by a single network. It fosters a decentralized ecosystem, **devoid of any dominant controlling entity**, by creating an **interconnected** layer with Demos, facilitating seamless data and functional transfer between networks. This interconnection allows for efficient resource exchange and utilization, promoting a **cohesive and synergistic ecosystem**.

Design and Principles

Cross-chain Execution refers to a novel **design principle** that facilitates efficient and rapid interaction between different blockchains. This advanced concept is particularly advantageous in scenarios where smart contracts or a collection thereof necessitate a substantial **computational workload**.

For instance, consider a dApp developer who desires to maintain the **security** of a gas-intensive chain like Ethereum while capitalizing on the **lower gas costs** associated with an L2 chain such as Binance Smart Chain (BSC). The introduction of Cross-chain Execution has made it possible to **execute portions of a smart contract** on BSC, while the ultimate outcome is validated and finalized on Ethereum. By adopting this approach, developers can harness the swiftness and efficiency of BSC while simultaneously upholding the security and functionality offered by Ethereum.

Furthermore, Cross-chain Execution empowers developers to create powerful dApps that exploit the unique strengths of multiple blockchain networks, **without confining themselves to a single chain.**

To enable the implementation of Cross-chain Execution, Demos has developed user-friendly standardized endpoints and software development kit (**SDK**) methods. These resources facilitate Cross-chain execution for both clients and developers. The platform's unified architecture adheres to the principle of **interoperability**, ensuring seamless collaboration among diverse blockchain networks.

Each supported chain **encapsulates its available methods**, such as establishing a connection with a provider or retrieving balances, within a **standardized object** that exposes identical methods across all supported chains. This standardized approach simplifies the execution of transactions across different blockchain networks, as developers can utilize a **single method, such as "Connect,"** which remains applicable to every supported chain.

Moreover, the platform's SDK functions furnish developers with a straightforward and intuitive **interface to interact with the blockchain network**, thereby streamlining the process of building decentralized applications.

The SDK undergoes **continuous maintenance and updates** to accommodate the expanding range of supported blockchains. Importantly, these enhancements are implemented **without necessitating developers and users to make significant changes to their existing code**, thereby minimizing the learning curve while preserving flexibility and simplicity.

Similarly, transaction execution adheres to the same underlying principles, bolstered by an additional layer of security that mandates authentication of the requester before executing any write-like operation. To avoid transmitting sensitive data such as private keys, Demos accepts **signed payloads** that can be relayed to the target network providing authentication and security **without exposing any private information.**

Securing XM Context

Cross-chain data transfer and thus interoperability is a very delicate field. Among the myriad of possible security problems, the main barrier for developers and clients is the non existent standardization of chain operations.

Demos implements the **Demos SDK XM module**. With the SDK, clients can seamlessly construct **multichain transactions** in an intuitive manner by leveraging SDK methods and already established languages such as TypeScript . The Demos RPC processes the XM Payload, receiving it and returning the results along with **attestation data** from all the involved chains.

While the aim of this process is to be as swift as possible, it's crucial to acknowledge that **the speed of a transaction is limited by the slowest network within the operation**. This holds particularly true when each transaction requires **finality**. However, for read-only transactions, this logic generally doesn't apply as they typically don't necessitate finality.

Read, Write, and Cross-chain Presence

To enable interoperability across various blockchains, Demos employs a comprehensive framework that supports two distinct types of operations within the domain of Cross-chain Execution.

The **Read** operations entail retrieving data from target networks **without requiring explicit authentication**, except for specific scenarios. For instance, examining the account balance on Solana does not necessitate possessing a Solana account, and this principle holds true for nearly all read operations across diverse networks. This operational category proves immensely valuable for **swiftly obtaining data** without the need to engage in time-consuming processes like handshake establishment, authentication, and data propagation associated with on-chain modifications.

The **Write** operations demand meticulous and **trusted authentication** on the target network to ensure secure execution. Since these operations involve altering data on-chain, utmost caution, efficiency, and discretion must be exercised to ensure data accuracy and identity

integrity. By leveraging **client-side signatures** when possible, Demos effectively serves as a **virtual RPC mechanism** for the target network, while concurrently exposing the aforementioned universal Demos methods. Consequently, Demos **does not require the user's private key** or sensitive information, instead acting solely **on behalf** of the requester for that specific request, utilizing the specific content provided by the user.

On the other hand, certain operations may necessitate a more direct approach wherein relayed transactions relying solely on signatures prove insufficient. In such scenarios, Demos assumes an intermediary role, executing logic directly on a blockchain while **soliciting additional requirements from the requester,** such as gas fees or specific information.

In both cases, **Demos is able to select the most efficient route** and deliver an intuitive, user-friendly, and privacy-focused solution that remains agnostic to the underlying blockchain, benefiting developers, users, and clients alike.

XM Security on Write Transactions

It's crucial to highlight a key aspect of the Demos SDK XM Module implementation: by default, the module doesn't transmit, demand, or in any way request sensitive information.

For Write transactions, whether they are EVM-related or involve transfers on other Layer 1 networks, the framework equips clients with all the **essential methods to generate and sign transactions** and payloads entirely on **the client side**. With the exception of the "Execute" step, which necessitates reachability to the RPC, it is entirely feasible to **construct a fully compliant XM Transaction offline.**

This methodology is referred to as "**Security through Ignorance**" and forms the foundation of the Demos Network's communication approach.

Here is an example of a possible XM Transaction workflow:

– The client initiates transaction building **entirely offline** by disconnecting their router.

– Upon realizing the need to use the Pay method on the EXAMPLE (XMPL) chain, the client utilizes the Demos SDK to **locally sign that operation**.

– Operations, both read and write, are continuously added to the transaction.

– Upon completion, the client possesses an object (a **XM Transaction**) containing all operations in their specified order.

– Write operations **include signed payloads** devoid of sensitive information.

– The client reconnects the router and **dispatches the request to Demos RPC.**

– Demos RPC employs the Demos XM Module to execute, relay, and manage the included operations without any modification.

– The resulting data is interpreted and processed, ensuring the client receives it in a readable format.

This way, **no sensitive data can be leaked in any part of the process.**

Web2 and Cross-Web Logic

As Demos is built around different features that can and will interact with each other, one of the derived functionalities, known as Cross-Web Execution, is a combination of Cross-chain Execution and **On-chain Web2**, with its technical explanation being the sum of these constituent elements. It is necessary to highlight this **synergistic** feature, which may not be immediately evident.

By employing the established methodologies for **on-chain retrieval of Web2 resources** with assured integrity and security, alongside **seamless and secure cross-chain** smart contract execution, the convergence of these approaches gives rise to unparalleled capability.

For instance, a dApp **can incorporate smart contract logic deployed on an EVM network while leveraging attested Web2 data as input**. Conversely, developers have the flexibility to **feed the results of a smart contract into a Web2 API,** such as an **AI endpoint.**

The potential applications are virtually **limitless**, contingent upon the performance characteristics of the involved endpoints. Consequently, the resulting data can be once again utilized in a similar manner across other networks, endpoints, or technological frameworks.

Example: Cross-chain Swapping

Cross-chain Swapping, as an implementation of Cross-chain execution, constitutes a distinctive use case that necessitates a dedicated exposition of its fundamental mechanics.

To facilitate a comprehensive understanding of Cross-chain swapping's underlying technical principles, it is beneficial to provide a brief explanation of this feature. Cross-chain Swapping entails the **seamless, rapid, and efficient execution of direct swaps between disparate blockchain networks**. For illustrative purposes, we shall consider a practical scenario where an ERC20 token is acquired using a Stablecoin on the Solana blockchain.

Design and Multiple Routes

During the development of Cross-chain swapping, it became evident that a single method could not consistently deliver the optimal combination of efficiency, speed, and security. Due to varying conditions on both sides of the swap operation and within the Demos network itself, Demos must possess the **intelligence to select the best overall rout**e from a set of possible options.

To address this requirement, we have devised two primary scenarios in which Demos can make decisions regarding the chosen path.

1. **Full Liquidity Availability:**

   In the ideal situation, Demos will possess sufficient liquidity (**for example through Liquidity Tanks**) on both sides of the swap, enabling it to p**romptly provide liquidity** after validating the swap request. In this case, Demos will either distribute the requested token directly to the buyer's address or, if necessary, exchange the available liquidity for the requested token before distribution. This method, which is the fastest and preferred approach, will be employed whenever the **Full Liquidity Availability** condition is met.

2. **Limited Liquidity Availability:**

   When Demos lacks the necessary liquidity on both sides of the swap, it acts as a **bridge and ramps aggregator** with enhanced security measures. This scenario is expected to occur during the initial stages of Demos implementation and upon integration of novel Blockchains.

   While maintaining the same interface and syntax for developers and users, internally Demos **scans and selects from a list of bridges (on-chain) and ramps (off-chain via Web2 endpoints) to identify the most efficient and secure route.** This eliminates the need for the requester to manually select a bridge or ramp, thereby saving time and minimizing the exposure of funds.

   In addition to this efficient selection process, Demos incorporates a **firewall layer** that continuously and heuristically **detects security issues on bridges and ramps** as quickly as possible. The mechanism employed by the firewall operates in diverse ways, which will be explained in the following paragraph.

Demos Firewall

In the context of the Limited Liquidity Available scenario, as previously mentioned, Demos employs a sophisticated mechanism to identify and select the optimal bridge or ramp for providing a seamless, efficient, and expeditious experience to the requester. As widely acknowledged, bridges represent a prominent target for malicious actors seeking to exploit vulnerabilities and steal funds. To counteract this threat, Demos incorporates a firewall layer.

Constant Auditing

The Demos network employs a perpetual and automated process to **audit the smart contracts and components of ramps and bridges**, meticulously scrutinizing them for common vulnerabilities, exploits, and attacks. Despite variations in code implementation, it is noteworthy that a majority of bridge exploits stem from **similar types of vulnerabilities**. Demos vigilantly monitors the security posture of the platform to **proactively** thwart the utilization of insecure pathways, all while remaining imperceptible to the end user.

Immediate Response Propagation

In instances where a bridge or ramp falls victim to exploitation, the attackers' primary source of gain does not solely derive from the initial attack. The most substantial harm is inflicted upon users who, **oblivious to the service's compromise,** continue utilizing it for prolonged periods, sometimes spanning **hours or even days**, until they become aware of the situation.

Within this realm, Demos assumes the role of an all-encompassing, automated monitoring service. Even in cases where an exploit employs ingenious tactics that evade immediate detection, Demos leverages **heuristic analysis of transactions**, news reports, and official statements to swiftly identify the issue and impede vulnerable routes for all Demos users, without them noticing it.

Leveraging the aforementioned techniques, the Demos Firewall manifests as an advanced system that integrates heuristics, analytics, and machine learning, effectively safeguarding users, even when operating as an intelligent aggregator of bridges and ramps.

L2 Parallel Subnetwork (L2PS)

While the Demos Network is inherently fast and cheap, it still has to face some of the common problems that blockchain-like networks encounter in providing data.

Due to the transaction gas framework that is critical to maintain a clean, safe and performant network, some kinds of data exchange would require an impractical amount of gas and an unacceptable latency.

A classic example, which we will examine in detail later, is Instant Messaging. **As of today, there are no fully on-chain mainstream solutions offering instant messaging in a practical form.**

Before diving deep into this example, though, Demos L2PS has to be introduced.

What is a Subnetwork?

In the context of Demos Network specifications, **a subnetwork is a network of Demos peers that in addition to the Demos main specifications integrate optional specifications defined in this paper.**

The integrated and automatic subnetwork included in any Demos node client is the **Instant Messaging Protocol (IMP**) which is used intensively by the Demos Network to efficiently deliver communications without clogging the main chain.

Why Parallel?

Any L2PS integration in Demos must be at the very least fully compatible with a running Demos node and its pre-built modules. This means that **no L2PS subnetwork can be run without a full Demos node installed and running.**

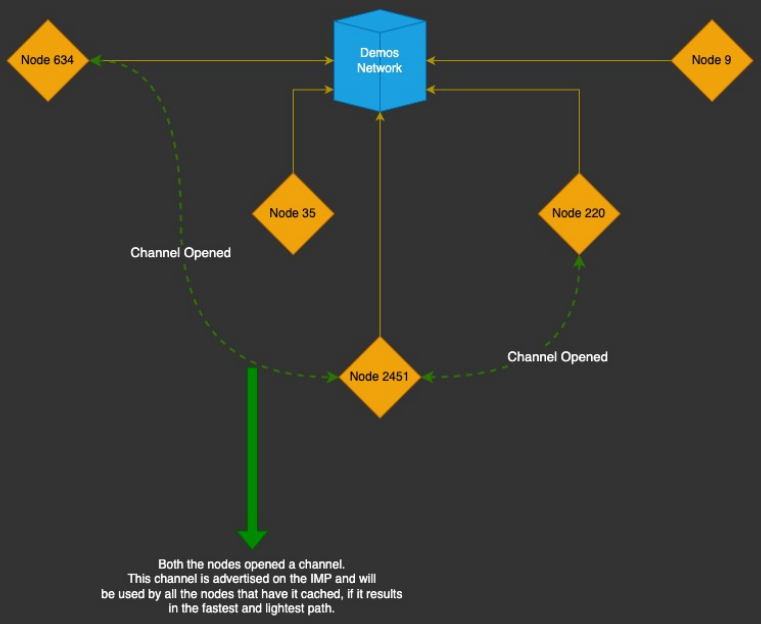
The “parallel” definition indicates that L2PS **are not, as the usual L2 concept may hint, separate networks** that catch up with an L1: they are, instead, parallel protocols and channels that are used to provide resources and services in the most efficient way possible.

IMP: Demos built-in L2PS

Instant Messaging Protocol

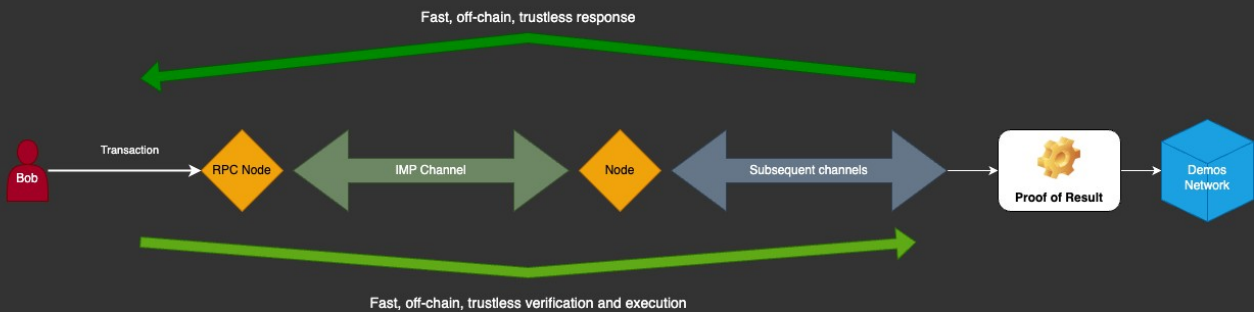
To keep most of the load off-chain, Demos Network implements a P2P network layered on top called IMP (Instant Messaging Protocol).

By opening direct communication channels between themselves, enough nodes are able to keep track of a workflow (for example an instant messenger), witness each other and submit a proof of result on Demos once the flow has finished.



Practical Example

Bob wants to execute a given transaction (any valid Demos transaction). Its transaction instructs the RPC to use the IMP network when possible. All the workflow that is able to be channeled in the IMP will be executed, verified and computed off chain by using the same Demos trustless cryptography. Only the final, cryptographically secure result is then stored and attested on chain. The savings using IMP can reach 95% of the TX size.



As mentioned above, Demos specifications define a **mandatory L2PS called IMP**. As the diagram suggests, IMP is designed to **reduce network load** whilst providing an efficient communication channel that is usable in different ways.

IMP needs to create **direct communication channels**, as defined in Demos specifications, with near peers. While this is usually one of the weaknesses of an L2 network, in the L2PS scenario **the IMP node is also the Demos node** and thus inherits all of the peers from the beginning.

By building a dynamic, verifiable and unpredictable chain of IMP nodes, the L2PS is able to quickly and safely certify many types of data that don't require immediate finality or that don't require finality at all.

As usual, an **example** explains it better:

- Alice wants to start a chat session with Bob, and uses the Demos SDK to call the proper method
- The Demos SDK connects to the Demos RPC and sends the L2PS request
- The Demos RPC interprets the L2PS request and verifies which operation is requested
- To find Bob, the Demos RPC uses the already established IMP network of peers to propagate the request "Where is Bob?"
- Due to the redundancy of the IMP connections, Bob is found almost immediately
- Alice's RPC either connects to Bob's current RPC (if Bob is connected and waiting for Alice), or writes the message in Bob's inbox
- Bob is able to either immediately receive or fetch the message
- Routine garbage management ensures the healthy condition of IMP L2PS

While this example explains in detail how L2PS logic works through an example of an IMP transaction, it also creates numerous questions that require a proper explanation. In the next paragraphs, we will examine each one of them.

Privacy and L2PS: A built-in concept

In the example above, the notable observation is the complete omission of any mention of privacy and sensitive information.

The Demos Network has built in privacy focused modules and integrations, for example the **Homomorphic Encryption** module. This module is consistently updated, maintained, and applicable to various facets of Demos transactions. Apart from the chat example which would be secured through **End-to-End encryption**, there are many scenarios where an actor, to safeguard privacy, necessitates not only the temporary absence of data persistence but also the heightened privacy offered by Homomorphic Encryption.

Homomorphic Encryption: Tailoring Privacy for Real Use Cases

To establish a robust privacy framework, Demos acknowledges the necessity of a novel approach to Zero-Knowledge operations, complementing traditional **ZK Proofs available through integrations**. Consider a scenario where a user wishes to transmit encrypted data without disclosing it, yet enabling others to perform trusted and verifiable operations on the data.

This approach is known as **Homomorphism**, and it involves the use of Homomorphic Encryption for secure data manipulation. Specifically, **Fully Homomorphic Encryption (FHE)** empowers Demos users and developers to create cutting-edge, privacy-preserving applications on the Demos Network. FHE ensures that users can enjoy the benefits of advanced privacy features without compromising on dynamism and functionality.

A Practical Example of FHE

Given the recent developments in Fully Homomorphic Encryption (FHE) and Homomorphic Encryption in general, it's helpful to explore a potential use case. Consider the scenario where Alice possesses a confidential number and wants Bob to add his data to this number without revealing the specifics.

Traditionally, Alice might ask Bob for the sum, verify its authenticity, manually add it, and then provide Bob with a proof of the completed operation. However, with an FHE approach, these steps can be streamlined into distributed processes:

- Alice encrypts her sensitive variable containing the number 5 using Homomorphic Encryption within the Demos SDK
- Bob, receiving the encrypted message, is unable to decipher its content
- Using the Demos SDK FHE module, Bob adds his value (2) to the encrypted message **without needing to decrypt**
- Alice, receiving the modified and encrypted data, decrypts it using the Demos SDK, as she originally encrypted it
- The decrypted data reads 7 (5+2), and Alice can cryptographically verify the result with security measures in place

A Trustless Approach to Data Transformation

The simplified example above serves as an illustration of how **privacy is intricately woven into the user experience** within the Demos Network specifications. Emphasizing the significance of employing suitable technology for specific features, Fully Homomorphic Encryption (FHE) introduces a rapid and focused layer of security across the entire network.

APPENDIX 1 – Glossary

Core Architecture Terms

Context A domain or environment in which data sources can be found. Examples include Web2 (traditional internet services) and XM (Cross-chain Media, referring to blockchain networks).

DAA (Decentralized Adaptive Architecture) Demos' overall architectural model designed to achieve efficiency, scalability, and performance through dynamic system adaptation and representative sharding.

Demos Network The complete blockchain system including all nodes, protocols, and specifications that enable cross-chain and Web2 integration.

GCR (Global Change Registry) A comprehensive ledger that maintains the state of the entire blockchain network, allowing transactions to be validated before block emission.

Node An entity that follows Demos specifications by providing a client/server architecture compatible with other network nodes.

RPC (Remote Procedure Call) Endpoints that handle transaction validation, relay, and processing within the Demos network.

Consensus and Validation

BFT (Byzantine Fault Tolerance) A consensus mechanism that enables agreement even when some nodes are malicious or faulty, requiring supermajority (typically 2/3+1) agreement.

CVSA (Common Validator Selection Algorithm) A deterministic algorithm that ensures all legitimate nodes select the same validators for consensus without requiring communication between nodes.

PoR-BFT (Proof-of-Representation with Byzantine Fault Tolerance) Demos' two-layer consensus mechanism combining validator selection through PoR with BFT consensus within shards.

Representative Shard The group of validators selected through CVSA to participate in a specific consensus round for block validation.

Shard A subdivision of the network that processes transactions independently to improve scalability and performance.

Validation Shard See Representative Shard. The terms are used interchangeably in the document.

Cross-Chain and Web Integration

DAHR (Data Agnostic HTTP(S) Relay) A proxy system that enables secure communication between blockchain clients and Web2 resources while maintaining integrity verification.

DTR (Decentralized Transaction Relay) The mechanism by which transactions are distributed across network nodes to ensure efficient processing and validation.

L2PS (L2 Parallel Subnetwork) Optional parallel protocols that run alongside the main Demos blockchain to provide additional services without congesting the main chain.

Liquidity Tank Smart contracts or wallets on different networks that hold funds to facilitate cross-chain transactions and gas fee management.

Native Bridge Built-in functionality that enables secure value transfers between different blockchain networks without external bridge protocols.

XM (Cross-chain Media) Operations that involve multiple blockchain networks, encompassing both read and write operations across different chains.

Technical Components

E2E (End-to-End) Validation process that verifies the authenticity of all parties involved in a transaction before data exchange.

FHE (Fully Homomorphic Encryption) Cryptographic technique that allows computations to be performed on encrypted data without decrypting it.

IMP (Instant Messaging Protocol) The mandatory L2PS included in all Demos nodes to provide efficient communication channels.

ML-DSA Module-lattice based digital signature algorithm (formerly Dilithium) used for post-quantum cryptographic security.

ML-KEM Module-lattice based Key Encapsulation Mechanism used for quantum-resistant key establishment.

PQC (Post-Quantum Cryptography) Cryptographic algorithms designed to be secure against attacks by quantum computers.

Network Operations

Client Software or application that interacts with the Demos network through RPC endpoints.

Epoch A time period during which a specific set of validators is responsible for consensus operations.

Finality The point at which a transaction is considered irreversibly confirmed on the blockchain.

Intent A user's expressed desire to perform specific operations, which the network then executes deterministically.

Mempool The collection of pending transactions waiting to be included in the next block.
SDK (Software Development Kit) Tools and libraries provided by Demos to enable developers to build applications that interact with the network.