INITIUM WHITEPAPER; VERSION INANIS

A proposal for a Web-Scale Blockchain for fast, secure, scalable dApps and marketplaces.

VERSION: INANIS
JUNE 2022

Initium Foundation
labs@initium.foundation

Legal Disclaimer

This whitepaper describes the Initium version Inanis, a.k.a. Iminium Inanis. Nothing in this Whitepaper is an offer to sell or the solicitation of an offer to buy any tokens. Initium Foundation (a.k.a. Initium) is publishing this Whitepaper solely to receive feedback and comments from the public. If and when Initium offers for sale any tokens (or a Simple Agreement for Future Tokens), it will do so through definitive offering documents, including a disclosure document and risk factors. Those definitive documents also are expected to include an updated version of this Whitepaper, which may differ significantly from the current version. If and when Initium makes such an offering in the United States, the offering likely will be available solely to accredited investors. Nothing in this Whitepaper should be treated or read as a guarantee or promise of how Initium's business or the tokens will develop or the utility or value of the tokens. This Whitepaper outlines current plans, which could change at its discretion. The success will depend on many factors outside Initium's control, including market-based factors and factors within the data and cryptocurrency industries. Any statements about future events are based solely on Initium's analysis of the issues described in this Whitepaper. That analysis may prove to be incorrect.

About Initium Foundation

Initium Foundation (a.k.a. Initium) is a non-for-profit decentralized autonomous organization (DAO) for developing and promoting the ecosystem of the Initium blockchain project. Members are free to join the Initium Foundation with the commitment to participate in the development and promotion of the project. Please visit https://initium.foundation for further information.

Abstract

This White Paper proposes a new blockchain architecture based on Initium's consensus, Proof of Synchronization (PoSync), also called "Covenant"- a proof for verifying order and synchronization of time between events. Covenant is used to encode the trustless passage of time into a ledger- an append-only data structure. When used alongside a consensus algorithm such as Proof of Replication (PoRep) or Proof of Stake (PoS), Covenant can reduce messaging overhead in a Byzantine Fault Tolerant replicated state machine, resulting in sub-second finality times. This paper also proposes two algorithms that leverage the time-keeping properties of the Covenant ledger-the PoS algorithm that can recover from partitions of any size and an efficient streaming PoRep. The combination of PoRep and Covenant provides a defense against forgery of the ledger concerning time (ordering) and storage.

Version Inanis

This Whitepaper describes the main ideas behind the Initium Inanis (version Inanis). Inanis Version of Initium is the test version of the Initium blockchain which is to be considered the lab version. The codes of this version are available on Initium Foundations GitHub. The next version is the Initium Primus and its release date will be announced by the Initium Foundation. The Primus version will enable the test net of the Initium blockchain. The successful tests of Initium Primus would lead us to release the main version, the Initium Genesis.

Table of Contents

1. Introduction

Developing a new blockchain architecture by considering Vitalik Buterin's trilemma is a challenging task as the developers face a set of three significant challenges, decentralization, security, and scalability. It's widely believed that the blockchains are built to force developers to sacrifice one of the aspects in favor of the two, as they can only provide two of the three benefits at any given time.

Although scalability is an essential issue for broader adoption of the blockchain technology, especially for micropayments and dApps, the developers and users need an optimum network that can provide them with three benefits of the blockchain.

Time synchronization among network partners is a fundamental requirement for efficient and reliable communication on a global scale. Typical synchronization schemes like NTP operate on a trust-based client-server model, which does not scale well in a decentralized network. Single server failures can lead to severe downtime before re-establishing synchronization.

Initium's consensus, the Proof of Synchronization (PoSync), also called Covenant, will provide a seamless solution for the synchronization of nodes throughout the network, which is highly cost-effective while keeping the decentralization, security, and scalability. It makes the Initium a potential network to be used by IoT machines, dApps, marketplaces, and micropayments.

The Initium project is a layer-1 blockchain protocol for smart contract developed based on the Rust and WASM. Initium Foundation, as a DAO, aims to develop the Initium ecosystem through the participation of developers and volunteers around the world. At the time of compiling this Whitepaper, the test codes of the project are available as an open-source project on GitHub. The developing team of Initium is currently working on developing SDK, wallet, and the different modules required to prepare the project for the test net.

2. Outline

Section 3 provides an understanding of the network design of the Initium's blockchain and the rules of the nodes on the network.

Section 4 explains the Covenant, the innovative consensus of the Initium's blockchain as the Proof of Synchronization (PoSync), and the mechanism to achieve the consensus throughout the network.

Section 5 describes the compatibility and adaptability of Covenant with Proof of Replication (PoRep) and Proof of Stake (PoS) as practical byzantine fault tolerance (pBFT) protocol.

Section 6 explains the system architecture of the Initium network and the communication of the state throughout the network.

Section 7 describes the Initium Virtual Machine (IVM) and its use cases for further works and studies.

3. Network Design

The Initium benefits from robust network architecture to enhance the network's scalability, security, and decentralization. As illustrated in Fig.1, at any given time, a system node is designated as Prime Node to generate a Proof of Initiation sequence, providing the network global read consistency and a verifiable time frame for verifying the transaction. The Prime Node sequences user messages and orders them such that they can be efficiently processed by other nodes in the system, maximizing throughput. It executes the transactions on the current state stored in RAM, publishes them, and sends a final state signature to the replication nodes called Verifiers. Verifiers execute the same transactions on their copies of the state and publish their computed signatures of the state as confirmations. The published confirmations serve as votes for the consensus algorithm. Verifiers receive transaction fees for the transactions their process and verify. In addition to the fees, the verifiers that participated in producing a block will also receive the block reward.

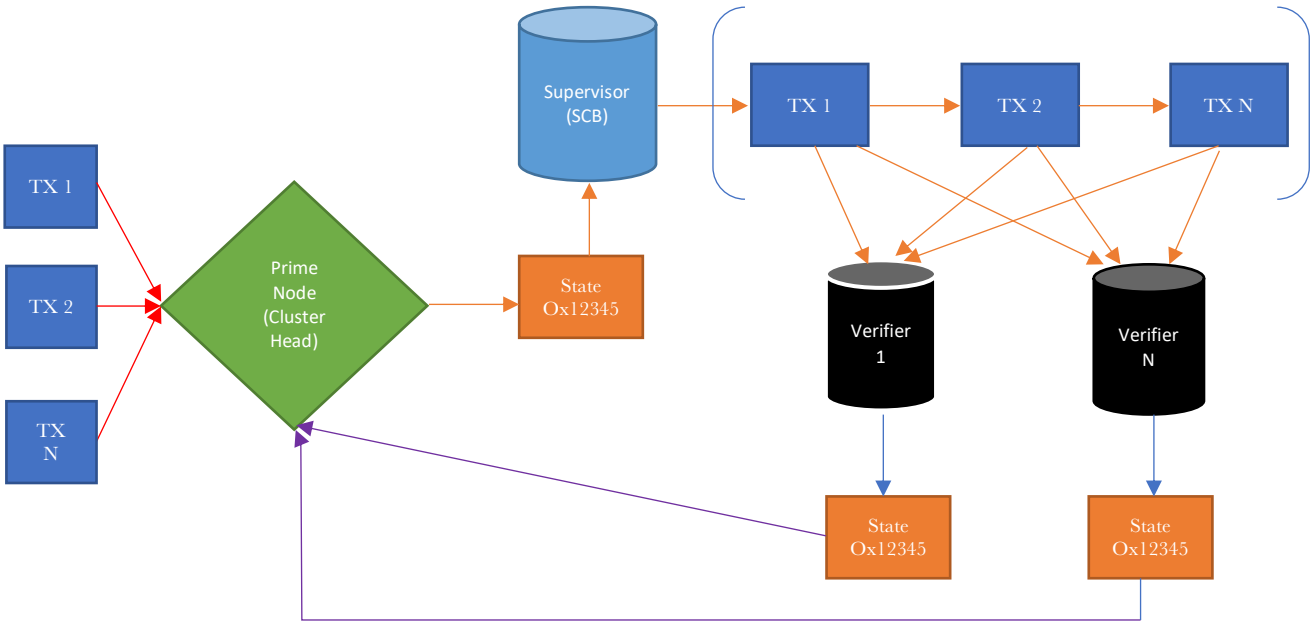


Figure 1. The flow of information and the interaction of CH, SCB, and CB nodes in Initium Network

The Prime Node rule is a temporary role in the network since the verifiers vote to choose the next Prime Node to set the new state. Every Verifier has the equal right to vote and the equal chance to be elected as the next Prime Node. This election takes place based on the PoS mechanism described in Section 5.

In terms of the CAP theorem, Consistency is almost always picked over Availability in the event of a Partition. In the case of a large partition, this paper proposes a mechanism to recover network control from a partition of any size.

4. Covenant

Covenant is a cluster-based synchronization protocol that uses clustering to achieve energy efficiency and resilience to faulty nodes in the system. Clustering techniques are known to reduce the energy expenditure of the nodes and provide routing through representative nodes called Cluster heads (CH). We refer this Cluster Head (CH) node in Covenant protocol as Prime Node. Since communication in Covenant (Proof of Synchronization) involves only Prime nodes, the number of messages exchanged is reduced significantly compared to other protocols. Our preliminary results show that Covenant achieves a worst-case synchronization error of 37 *μs* (0.037 *ms*) in a multi-hop network of up to 10 hops.

Covenant protocol operates in two phases of 1) *clustering* and 2) *consensus* and adopts a slotted operation mode where synchronization is performed in fixed time slots while applications are executed in the remaining time slots, as shown in Figure 2.

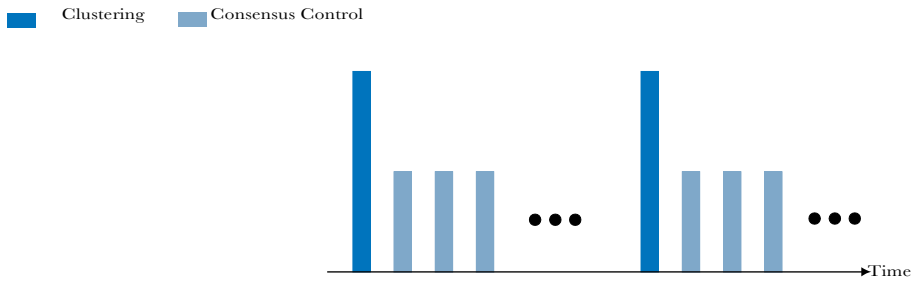


Figure 2. Covenant Protocol with two phases of operations

4.1 Clustering Phase

In the first phase of Covenant, the nodes are grouped into clusters. A clustering process is a state machine of 3 states: 1) neighbor discovery, 2) election, and 3) connection, as indicated in Figure 3.

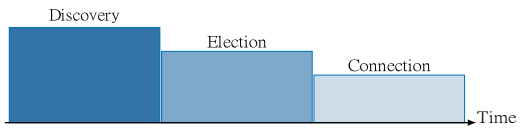


Figure 3. Covenant Protocol clustering phase with state transitions

Nodes become aware of their neighboring nodes in the neighbor discovery state. Time information is added to the messages to achieve an implicit synchronization during this state, similar to Gradient Time Synchronization Protocol (GTSP). GTSP achieves time synchronization by averaging neighbor time information and provides resilience for the clustering phase (C-GTSP). During the election phase, a representative node, the Prime Node, is chosen as a cluster head (CH) based on the maximum degree (number of connections) of a node. Lastly, the connection state selects a representative bridge node (nodes that belong to multiple clusters) as a cluster bridge (CB) and other bridge nodes as supervision cluster bridges (SCB) to ensure inter-cluster connectivity and resilience.

4.2 Consensus Phase

Consensus Control (Cons-Ctrl) is the second phase of Covenant which repeats periodically after specific time slots to maintain the synchronization. The cluster heads exchange time

information to find *local centers* of their neighborhood. The Local centers are CHs at a configurable number of hops from the network's edge.

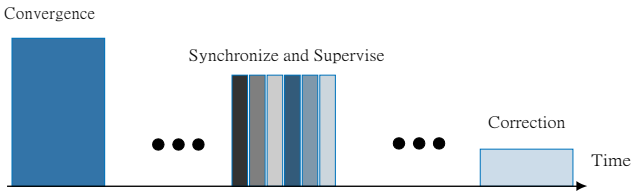


Figure 4. Convergenceand consensus phase with TDMA communication

The configuration limits the maximum number of hops before reaching the consensus. Local centers bound the maximum error across the network. The time information of the local centers is propagated back to the CHs in the reverse direction after the convergence. The order of the Prime Node receiving data is the reverse of finding local centers; hence, the time-division multiple access (*TDMA)* slots for each cluster are created accordingly for the *synchronize and supervise* phase. Thus, the protocol scales very well for more extensive network sizes without additional energy loss and additional delay for consensus. However, given the dynamic nature of the network, some nodes may fail, or new nodes may join the network. If there are significant changes (e.g., change in CH), clustering is triggered again to establish new clusters with the optional *correction* state of the consensus phase in Figure 4.

4.3 Detection of Byzantine nodes

The elected cluster bridge (CB) and supervision cluster bridge (SCB) nodes act as supervision nodes to monitor the broadcast data of Prime nodes to their cluster member nodes. SCB also ensures reliable and accurate information propagation during consensus. In the case of a byzantine fault, e.g., the Prime node sending corrupt time information, CBs and SCBs intervene and share the legitimate value of time. As Cluster Bridges are connected to other clusters, they further forward the valid time information along the network path. Hence, the impact of a byzantine event can be detected and contained within a single cluster.

4.4 Hashing Function

Covenant uses SHA256 algorithm to hash the data. Hashing new data starts with a cryptographic hash function, whose output is not predictable without running the function (e.g., SHAsha256). It runs the function from some random starting value, takes its output, and passes it as the input into the same function again. Record the number of times the function has been called and the output at each call.

PoSync Sequence

Index	Operation	Output Hash
1	sha256(“any random starting value”)	hash1
2	sha256(hash1)	hash2
N	sha256(hashN-1)	hashN

Where hashN represents the actual hash output.

As long as the hash function chosen is collision-resistant, the set of hashes can only be computed in sequence by a single computer thread because there is no way to predict what the hash value

at index N is going to be without actually running the algorithm from the starting value N times. Therefore, we can infer from the data structure that real-time has passed between index 0 and N.

4.5 Timestamp for Events

This sequence of hashes can also be used to record that some piece of data was created before a particular hash index was generated. Using a "combine" function to combine the piece of data with the current hash at the current index. The data can simply be a cryptographically unique hash of arbitrary event data. The combine function can be a simple append of data, or any operation that is collision resistant. The next generated hash represents a timestamp of the data, because it could have only been generated after that specific piece of data was inserted.

For example:

PoSync (Covenant) Sequence

Index	Operation	Output Hash
1	sha256("any random starting value")	hash1
199	sha256(hash198)	Hash199
N	sha256(hashN-1)	hashN

Some external event occurs, like a photograph was taken, or any arbitrarydigital data was created:

PoSync (Covenant) Sequence With Data

Index	Operation	Output Hash
1	sha256("any random starting value")	hash1
200	sha256(hash199)	hash200
1000	sha256(hash999)	hash1000
N	sha256(append(hashN-1, object sha256))	hashN

HashN is computed from the appended binary data of hashN-1 and the SHA256 of the object (e.g., photograph, contract, etc.). The index and the sha256 of the photograph arerecorded as part of the sequence output. So, anyone verifying this sequencecan then recreate this change to the sequence

4.6 Verification

The sequence can be verified correct by a multicore computer in significantlyless time than it took to generate it.

Core 1

Index	Data	Output
200	sha256(hash199)	hash200
300	sha256(hash299)	hash300

Core 2

Index	Data	Output
300	sha256(hash299)	Hash300
400	sha256(hash399)	Hash400

Given some number of cores, like a modern GPU with N cores, the verifier can split up the sequence of hashes and their indexes into N slices,and in parallel make sure that each slice is correct from the starting hash to the last hash in the slice. If the expected time to produce the sequence is going to be as followings:

$$\frac{\textit{Total number of hashes}}{\textit{Hashes per second for 1 core}}$$

The expected time to verify that the sequence is correct is going to be:

$$\frac{\textit{Total number of hashes}}{\textit{(Hashes per second for per core * Number of cores available to verify)}}$$

4.7 Horizontal Scaling

This is possible to synchronize multiple Prime Nodes by mixing thesequence state from each CH to every other CH, and thus achieve horizontal scaling of the Proof of Synchronization. As described earlier, Cluster Bridge (CB) nodes and Supervisor Cluster Bridge (SCB) nodes can easily handle this scaling and provide the end-to-end synchronization without sharding. The output of both generators is necessary to reconstruct the full order of events in the system.

Prime Node A		
Index	Hash	Data
1	hash1a	
2	hash2a	Hash1b
3	hash3a	
4	hash4a	

Prime Node B		
Index	Hash	Data
1	hash1b	
2	hash2b	hash1a
3	hash3b	
4	hash4b	

By periodically synchronizing the nodes, each node can handle a portion of external traffic. Hence, the overall system can take an enormous number of events to track at the cost of true-time accuracy due to network latencies between the nodes. A global order can still be achieved by picking some

deterministic function to order any events within the synchronization window, such as by the value of the hash itself.

5. Voting and Elections

Covenant uses Proof of Stake (PoS) for quick confirmation ofthe current sequence produced by the Prime Node, for voting and selecting the next Proof of Synchronization generator, and for punishing any mis-behaving validators. This algorithm depends on messages eventually arrivingto all participating nodes within a certain timeout.

5.1 Elections for New Prime Node

Election for a new Prime Node occurs when the Prime Node failure is detected. The validator with the most significant voting power or the highest public key address will be appointed as the new Prime Node if there is a tie.

A supermajority of confirmations is required on the new sequence. If the new leader fails before supermajority confirmations are available, the next highest validator is selected, and a new set of confirmations is required. For switching the votes, a validator needs to vote at a higher PoSync sequence counter, and the new vote needs to contain the votes it wants to switch. Otherwise, the second vote will be slashable. Vote switching is expected to be designed so that it can only occur at a height that does not have a supermajority.

Once a Prime Node is established, a Secondary node (SCB) can be elected to take over the transactional processing duties. If an SCB exists, it will be considered the next leader during a Primary failure.
The election is designed so that the SCB becomes Prime and the lower rank nodes are promoted if an exception is detected or on a pre-defined schedule.

5.2 Elections for SCB Node

Secondary and lower-ranked nodes, including SCB Node, can be proposed and approved. A proposal is cast on the Prime Node sequence. The proposal contains a timeout. If a supermajority approves the motion of the vote before the timeout, the Secondary is considered elected and will take over duties as scheduled. The Primary Prime Node can do a soft handover to Secondary by inserting a message into the generated sequence indicating that a handover will occur or by inserting an invalid state and forcing the network to fall back to Secondary. If a Secondary is elected and the primary fails, the Secondary will be considered the first fallback during an election.

6. Replication Mechanism

Covenant uses as Fast Proof of Replication. The mechanism used in Covenant is not used as a consensus algorithm but is a helpful tool to account for the cost of storing the blockchain history or state at a high availability.

Without key rotation, the same encrypted replication can generate cheap proofs for multiple Proof of History sequences. Keys are rotated periodically, and each

replication is re-encrypted with a new key that is tied to a unique Proof of History sequence.
Rotation needs to be slow enough that it's practical to verify replication proofs on GPU hardware, which is slower per core than CPUs.

Proof of History generator publishes a hash to be used by the entire network to encrypt Proofs of Replication and used as the pseudorandom number generator for byte selection in fast proofs. Hash is published at a periodic counter roughly equal to ½ the time it takes to encrypt the data set. Each replication identity must use the same hash and use the signed result of the hash as the seed for byte selection or the encryption key.
Each replicator's period must prove that it must be smaller than the encryption time. Otherwise, the replicator can stream the encryption and delete it for each proof.

Every Initium node is not expected to validate the submitted Proof of Replication proofs. It is expected to keep track of the number of pending and verified proofs submitted by the replicators' identities. A proof is expected to be verified upon the replicator can sign the proof by a supermajority of the validators in the network.
The verifications are collected by the replicator via the P2P gossip network and submitted as one packet that contains a supermajority of the validators in the network. This packet verifies all the proofs before the specific hash generated by the Proof of History sequence and can have multiple replicator identities at once.

7. System Architecture

7.1 Prime Node

The Initium benefits from robust network architecture to enhance the network's scalability, security, and decentralization. As illustrated in Fig.1, at any given time, a system node is designated as Prime Node to generate a Proof of Initiation sequence, providing the network global read consistency and a verifiable time frame for verifying the transaction. The Prime Node sequences user messages and orders them such that they can be efficiently processed by other nodes in the system, maximizing throughput. It executes the transactions on the current state stored in RAM and publishes the transactions and a signature of the final state to the replication nodes called Verifiers.

7.2 Verifier (Supervisor)

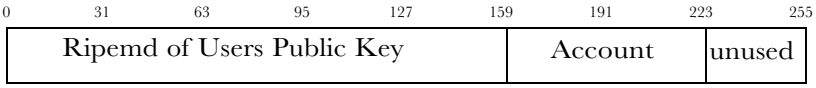
The Verifier nodes replicate the blockchain state and provide high availability of the blockchain state. The consensus algorithm selects the replication target, and the validators in the consensus algorithm select and vote the Proof of Replication nodes they approve of based on off-chain defined criteria. The network could be configured with a minimum Proof of Stake bond size and a requirement for a single replicator identity per bond.

7.3 Validators

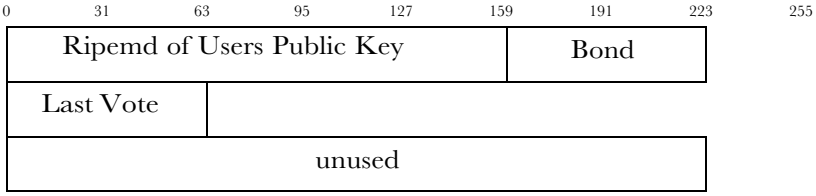
These nodes are consuming bandwidth from Verifiers. They are virtual nodes, and can run on the same machines as the Verifiers or the Prime, or on separate machines that are specific to the consensus algorithm configured for Initium.

7.4 State

State is a hash table indexed by the users address. Each cell contains the full address and the memory required for this computation. The example below shows a state for 32 bytes.

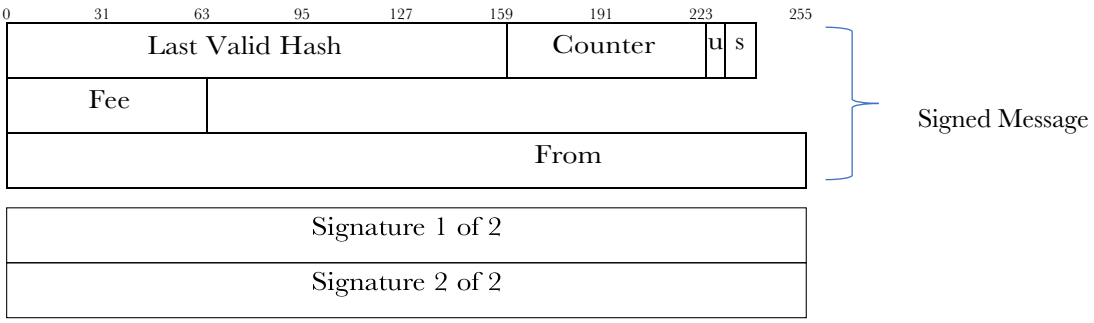


Proof of Stake bonds table contains a total of 64 bytes as the example below:

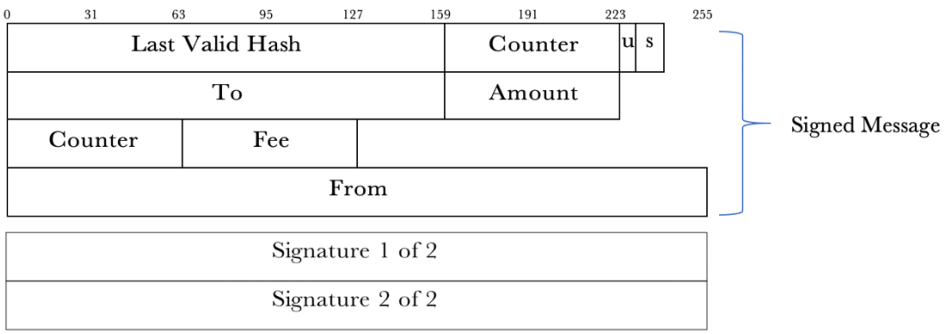


7.5 Incoming Pack Format and Network Limits

Prime Node is expected to be able to take incoming user packets, order them the most efficient way possible, and sequence them into a PoSync sequence that is published to downstream Verifiers. Efficiency is based on memory access patterns of the transactions, so the transactions are ordered to minimize faults and maximize prefetching. The incoming pack format with total size of 148 bytes is illustrated as below.

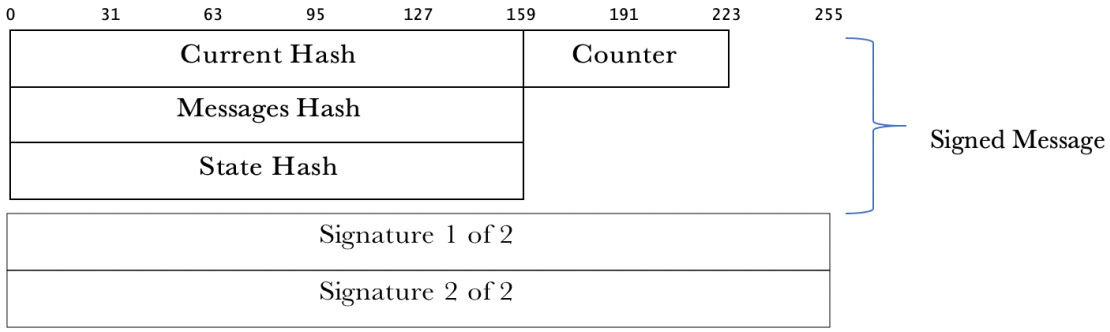


The minimal payload that can be supported would be 1 destination ac-count. With the following payload and minimum size of 176 bytes.



The Covenant sequence packet contains the current hash, counter, and the hash of all the new messages added to the PoSync sequence and the state signature after processing all the messages. This packet is sent once every N messages are

broadcast. The minimum packet size is 132 bytes.



On a 1gbps network connection, the maximum number of transactions possible is 1 gigabit per second / 147 bytes = 844k tps max. Some loss of a 4% is expected due to Ethernet framing. The spare capacity over the target amount for the network can be used to increase availability by coding the output with Reed-Solomon codes and striping it to the available downstream Verifiers.

7.6 Computational Limits

Each transaction requires a digest verification. This operation does not use any memory outside of the transaction message itself and can be parallelized independently. Hence, throughput is expected to be limited by the number of cores available on the system GPU-based ECDSA verification servers have had experimental results of 900k operations per second.

7.7 Memory Limits

A naive implementation of the state as a 50% full hashtable with 32-byte entries for each account would theoretically fit 10 billion accounts into 640GB. Steady-state random access to this table is measured at 1.1 107 writes or reads per second. Based on two reads and two writes per transaction, memory throughput can handle 2.75m transactions per second. This was measured on an Amazon Web Services 1TB x1.16xlarge instance.

8. Initium Use Cases

Successful uses cases are necessary for any new protocol for mass adoption. Initium is a highly potential layer-1 blockchain for development of an extensive range of uses cases. In this section provides some of these potential use cases.

8.1 Initium Virtual Machine

The Initium Virtual Machine (IVM) is a type of WebAssembly Machine (WASM) that enables developers to deploy and integrate their applications on Initium. The IVM, is compiled using Rust language and WASM.

Using Rust and WASM enables the developers to have serverless workflow. There are two prominent use cases for Rust and WebAssembly:

- Build an entire application — an entire web app based on Rust.
- Build a part of an application — using Rust in an existing JavaScript frontend.

The main function of IVM is to enable the high-performance smart contracts. In the Initium network, Smart contracts are a generalized form of transactions. These are programs that run on each node and modify the state. This design leverages extended Berkeley Packet Filter (BPF) bytecode as fast and easy to analyze and JIT bytecode as the language for the smart contract.
One of its main advantages is a zero-cost Foreign Function Interface. The Intrinsics, or functions implemented on the platform directly, are callable by programs. Calling the Intrinsics suspends that program and schedules the intrinsic on a high-performance server. Intrinsics are batched together to execute in parallel on the GPU.

For example, when two different user programs call the same intrinsic. Each program is suspended until the batch execution of the Intrinsics is complete. ECDSA verification is an example of intrinsic. Batching these calls to execute on the GPU can increase throughput thousands of times.
This trampoline requires no native operating system thread context switches since the BPF bytecode has a well-defined context for all the memory that it is using. eBPF backend has been included in LLVM since 2015, so any LLVM frontend language can be used to write smart contracts. It has been in the Linux kernel since 2015, and the first iterations of the bytecode have been around since 1992.

In addition, the Solang compiler can be used for converting any EVM compatible smart contract to IVM and WASM compatibles. Solang uses LLVM and as a Solidity complier can be used for Initium, Substrate, Solana, and EWASM.

8.2 Micropayments and DeFi

Initium is an ideal blockchain for developing and/or integrating the exiting micropayment networks as:

- Highly Scalable; the block time of Initium is approximately 1 second and every transaction can take place in 37 *μs (*0.037 *ms*).
- Low transaction fees: due to using highly cost-effective consensus, the Covenant, the cost of transactions on Initium can be incredibly lower than any other protocol.
- Anti-Censorship: The protocol can defend against this form of attack by dynamically adjusting how fast bonds become stale.
- Decentralized: Initium architecture prevents centralization as every node has the equal right for becoming the Prime Node and being a Prime Node is not a permanent position in the network.
- Secure: Initium is secure against long range attacks, ASICs attacks, and Spams.

8.3 Web 3.0

Initium can be used widely by the Web 3.0 applications due to its high-performance capabilities for smart contracts, decentralization, cost effectiveness, and security. Ranging from social media, payment systems, games, DAOs, wallets, etc., Initium can be a flexible and reliable blockchain for development and integration of Web 3.0 applications.

8.4 Non-Fungible Tokens (NFTs)

Initium can provide an efficient, secure, and cost-effective environment for developing various NFTs and NFT marketplaces.

8.5 Internet of Things (IoT)

The nature of Covenant consensus, Proof of Synchronization (PoSync) and its algorithm enables the IoT machines to be highly synchronized together at a global scale without the need for a centralized timing system. Decentralization of IoT devices is a futuristic approach that can provide the users with higher security, lower costs, and enhanced privacy.

9. Links

Initium Foundation official links are as follows:
Web: https://initium.foundation
GitHub: https://github.com/InitiumPrime
Twitter: https://twitter.com/initiumchain
Telegram: https://t.me/initiumchain

Additional information and details will be provided in the upcoming versions of this whitepaper.