Ansiblock, White Paper (version 0.1)

Authors:

George Katsia gkatsia@ansiblock.io
Shota Gvinepadze sgvinepadze@ansiblock.io
Maxim Lebedev mlebedev@ansiblock.io

1. Ansiblock Introduction

1.1. Abstract

Evolution of data communication technologies and increased global Internet availability accelerated the growth of thousands technologies for web, mobile and desktop application development. Today's application are running in classic "client-server" or "partially distributed" mode heavily depending on the functionality of underlying technologies.

Client Server

Partially Distributed

Decentralised

1st generation blockchain - Bitcoin initiated the use of blockchain technology for digital currency operations. Blockchain technology development led by Ethereum gave a birth to 2nd generation blockchain technology concept where blockchain can be used as underlying layer for software application than can be run in fully decentralized way.

Running applications in decentralized way may result with big technological disruption and spark many innovations in different industries. Despite big expectations from new digital economy growth, crypto startups developing decentralized application are facing huge challenges. Blockchain technology as platform for running decentralized applications is in embryonic stage and they are missing technical capabilities for massive adoption and development of distributed application. Blockchain technology providers today do not provide enough capacity neither for transaction processing speed (TPS) nor for decentralized data storage (DDS).

Ansiblock is innovative technology based on state-of-art algorithmic improvements. By introduces High Speed Messaging Pipeline, Smart Contract Engine, Ansiblock Consensus Algorithm and Ansiblock Distributed Data Storage Ansiblock strives to provide blockchain as a convergence place for people, machines and applications.

Ansiblock delivers:

- ➔ Byzantine fault tolerant blockchain
- ➔ High Speed 1,024,000 TPS on a single shard (1 Gbps network, with minimal packet size of 122 bytes)
- ➔ Low latency and finality
- ➔ Smart Contract implementation
- ➔ Privacy layer for private transactions
- ➔ Distributed Data Storage

Ansiblock for developers:
Ansiblock provides developers with language independent development platform for running Decentralized Applications and communication protocols for inter-blockchain operability.

1.2. Current Blockchains Platforms

1.2.1. Overview

The quiet blockchain revolution forces many companies to start technology research and development. There are several projects with commitments to deliver new generation blockchains. Ansiblock team performed analysis of currently most promising blockchain startups. We delved into their source code, white paper, testnet and mainnet, development roadmap and other publicly available information sources to examine and find out unbiased information related to: a) Transaction Throughput, b) Node Connection Speed, c) Latency* d) Finality**, e) Block Structure, f) Scaling Approach, g) Consensus Algorithm, i) Transaction Pricing.

***Latency** the time it takes from the creation of a transaction until the initial confirmation of it being accepted by the network (and how the confidence of acceptance increases over time).*

****Finality** - the property that once a transaction is completed, there is no way to revert it (or alter it). Basically, it's the moment when the parties involved in the transfer can consider the deal done. Finality can be deterministic or probabilistic.*

1.2.2. Competitors

While doing competitor solution examinations we have found that most of blockchain startups have a significant delay in product delivery compared to initial roadmap. New generation blockchain projects are in deep development stage and are facing challenges related to decentralisation, security and scalability. The real performance parameters significantly differ from initial commitments or offerings.

Project	**Ansiblock**	**Ethereum**	**Neo + NOs**	**Tron**	**IOTA**	**Nuls**	**Zilliqa**	**IoTex**	**EOS**
Consensus Algorithm	PoS, PoO, PoRep, PoSt	PoW + PoS	dBFT	dPoS	DAG, PoW	PoC	PoW, PBFT, Collective Signing	RDPoS	dPoS
TPS (Currently / Promised)	450k/1m*	15/?	1k/100k	1.5k/10k	1k/1.5k	800/1m	2.5k/?	1k/?	3k/1m
Block time (sec)	0.01-0.05	10-20	20-70	15	60-180	10	30	1	0.5
Transaction fee (USD)	Not Defined	0.3	Free	0.000001	Free	0.001	Not Defined	Not Defined	Free
Smart Contract	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DApps	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
SDK language	Multiple	Solidity	Multiple	Solidity	Abra	Java	Scilla	Solidity	C/C++
Scaling	No	Sharding	No	No	No	Parallel Multi Chains	Sharding	Sub Chains	No

Table 1: Competitor Comparison

Ethereum looks most mature blockchain platform nowadays but implementation of Proof-Of-Stake algorithm (efficient algorithm for proportionally assigning block rewards to token holders) and implementation of sharding as scaling solution will significantly prolong Ethereum development roadmap. Existence of legacy code, migration/upgrade process, implementation of complex cross-sharding functionality may also influence the product delivery timeline.

**With 1gbps connection bandwidth. For 2gbps connection Ansiblock network can achieve speed of 2,048,000 TPS*

1.2.3. Number of transactions in Ethereum Network

High speed processing of transactions in secured way is a major bottleneck for decentralised application migration to blockchain platform. Current capacity of Ethereum is 15 TPS, compared to 500,000TPS on NASDAQ. Despite of limited capacity we can see the growing demand on transaction processing on Ethereum network, with the average transaction fee around $0.2 - 0.3 (as of 11/11/2018).

Lowest number of **1329** Transactions On Sunday, August 9, 2015
Highest number of **1349890** Transactions On Thursday, January 4, 2018

Ethereum competitors like NEO, EOS, TRON, IOTA claim that they will have almost no transaction fees in the beginning, but they are going to introduce incremental fee structure in the passage of time. There are projects that claim they will have zero transaction fees, but they plan to get profit in a different way by: application launch, smart contracts executions or other ways.

1.3. Global Transaction Demand

We see future blockchains as a convergence point of various transactions generated by people, devices and AI applications. According to various researches (by Gartner, Forrester and others) number of connected devices are constantly growing:

Year	World Population	Connected Devices	Average Number of Devices per Person	Device Type
2020	7.6 Billion	50 Billion	6.58	Smartphones, Smart Home, Tablets, Computers, Wearable Devices, Cars and other devices.
2015	7.2 Billion	25 Billion	3.47	Smartphones, Tablets, Computers, Wearable Devices
2010	6.8 Billion	12.5 Billion	1.84	Smartphones, Tablets, Computers
2003	6.3 Billion	500 Million	0.08	Phones, Computers

Table 2: Number of Connected Devices (Source Cisco.com)

Use of blockchain platforms for transactions processing will significantly increase the performance requirements and also will influence on functional maturity of these systems.

1.4. Challenges

Major challenges innovative blockchain technology is facing today is providing high performance and the same time ensuring it with the security and immutability of the network.

Performance
We classified the most popular approaches of different projects to provide high speed transaction throughput:

- Off-Chain Transaction Processing
- Multiple Blockchains
- Sharding

With Off-Chain Transaction Processing (Bitcoin, Litecoin, Stellar) transaction are processed not only by blockchain and additional security and traceability issues rise. Multiple blockchains (Sparkster) implementation introduces complexity related to cross-chain transactions, cross-chain account data management, increased processing time and security issues. Sharding seems most promising solution for scaling but complexity of cross-shard transaction and new security issues makes it less attractive for increasing overall blockchain TPS throughput,

Ansiblock plans to provide 1,024,000 (with 1gbps internet connection bandwidth) TPS performance over classical blockchain node network with Ansiblock Proof-Of-Stake Consensus Algorithm. Ansiblock **is not using** Off-Chain Transaction Processing, Multiple Blockchains or Sharding for provision of 1,024,000 TPS performance.

Security
Building a high performance blockchain network populated with numerous number of nodes with ability to execute smart contracts gives a space to innumerable attack vectors. Ansiblock provides security against all known attack types like: Double Spending Attack, Grinding Attack, Transaction Denial Attack, Desynchronization Attack, Eclipse Attacks, Long-range Attack, Nothing at Stake Attack, Sybil Attack, Outsourcing Attack, Generation Attack, etc.

1.5. Mission

Amplify the use of distributed technologies by introducing feature rich blockchain with high performance and improved security.

1.6. Vision

Ansiblock strives to provide blockchain technology as a convergence platform which will introduce unified interface for microtransactions processing, data storage and analysis.

1.7. Industries and Applications

Ansiblock blockchain features could be leveraged by many industries and applications.

Industries

Finance	**Insurance**	**Oil & Gas**	**Healthcare**
Manufacturing	**Robotics**	**Transportation**	**Logistics**
Retail	**Utilities**	**Telecom**	**Government**
Pharmaceuticals	**Education**	**Social Services**	**Renewable Energy**

Table 3: Industries which can be operated by the Ansiblock network

Applications
Ansiblock high performance throughput with sub-second transactional latency and finality makes it an attractive platform for many applications. Moreover by introducing data storage and data analysis interfaces on the blockchain together with inter-chain messaging protocols will significantly increase number of applications use cases on Ansiblock platform. Ansiblock platform can be used for the following applications:

Security	Surveillance	Order Management	M2M Data Transfers
Digital Asset Management	Digital Asset Exchange	Gambling	Payment Networks
Supply Chain	Delivery Tracking	Fleet Management	Inventory Management

Table 4: Applications of the Ansiblock platform

2. Ansiblock Architecture

2.1. Architectural considerations

Ansiblock architecture was designed to disrupt current limitations of blockchain technology and contribute to amplification of decentralised application development, having in mind following considerations:

Performance and Scalability:

- ➔ Provide performance required for people running billions of devices and applications.
- ➔ Provide low (less than 1 second) latency and finality for blockchain transactions.
- ➔ Provide fast Ansiblock Consensus Algorithm based on Proof-of-Stake.
- ➔ Prodive Decentralised Data Storage over blockchain for storing files and user data objects.

Smart Contracts:

- ➔ Provide Smart Contract implementation supporting multiple Virtual Machines
- ➔ Provide Smart Contract implementation supporting eBPF and WebAssembly Virtual Machines
- ➔ Provide “Almost-Turing-Complete” Smart Contract implementation*

Interoperability:

- ➔ Provide Protocols/API for inter-blockchain transactions.
- ➔ Provide Protocols/API for inter-blockchain assets exchange operations.

Privacy & Security:

- ➔ Provide “all-known-attack” resistant blockchain with secure Ansiblock Consensus Algorithm
- ➔ Provide Private Layer for encrypting/decrypting user transactions and data.

Portability:

- ➔ Provide node running under major desktop/server environments.
- ➔ Provide wallets and applications for mobile and IoT devices.

Manageability:

- ➔ Provide tools for network monitoring and management.
- ➔ Provide node version control and instant update functionality.

Developer Oriented:

- ➔ Provide language independent Smart Contract development platform supporting C, C++, C#, Crystal, Java, Javascript, Objective-C, Python, Ruby, Rust, Scala, Swift languages.

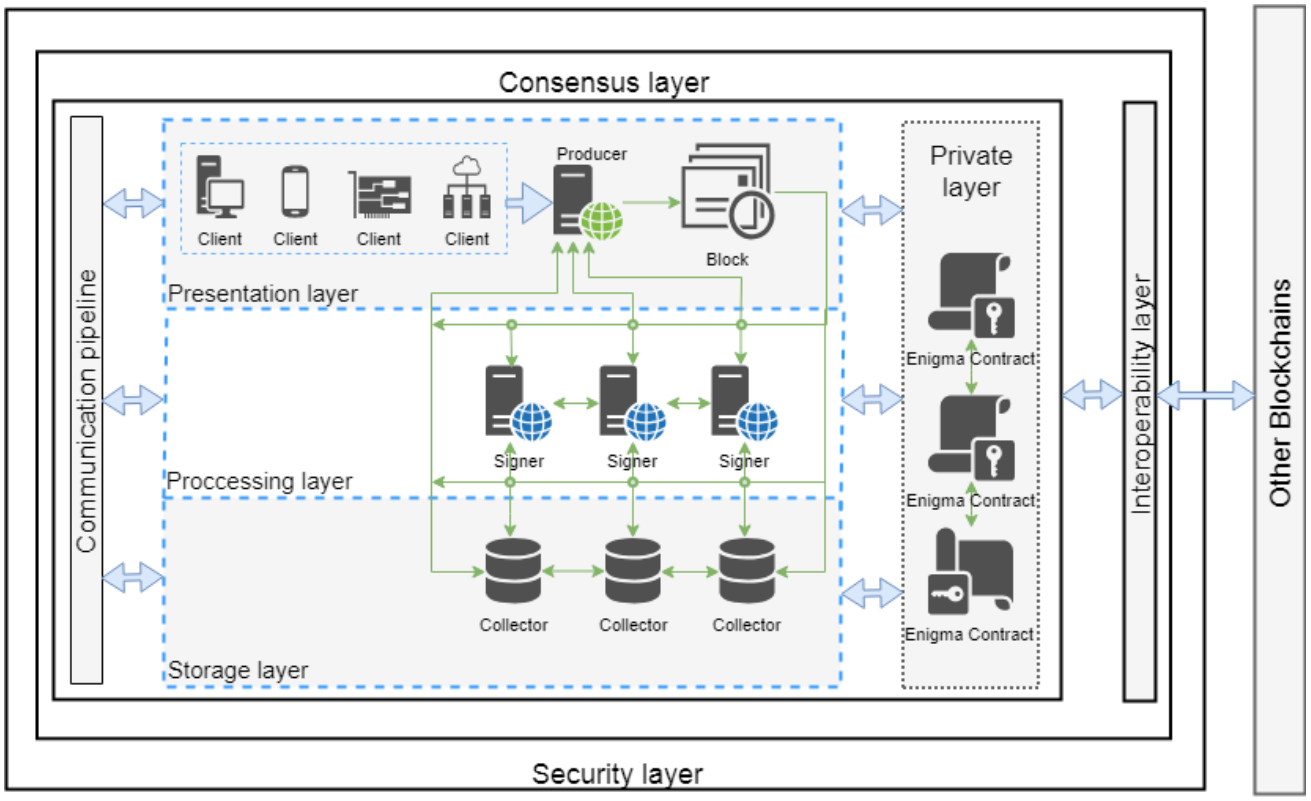
---

* A Turing machine is a mathematical model of computation that theoretically can perform any calculation that any other programmable computer can. The problem here lies in principle of the mathematical model for a Turing machine which in theory includes an unlimited amount of memory and no limit on how long a computation can take (as long as it finishes in a finite amount of time.) It is a known fact that real-world machines and programming languages do not have access to an infinite amount of memory, and practical programs must finish their work in a reasonable amount of time. So, regardless of some projects and developers claims about having a turing complete implementation, the fact is there are basically no turing complete solutions in the world because of halting problem.

2.2. High Level Design

High level design diagram of our blockchain represents logical interconnection of Ansiblock network functional components:

- Presentation Layer
- Processing Layer
- Storage Layer
- Security Layer
- Consensus Layer
- Interoperability Layer



Pic.1 High-Level Design Diagram

2.3. Block and Transaction Structures

Ansiblock blockchain introduces the following block structure, with max block size 64 kB:

Field Name	Size (bytes)	Description
block_height	8	Block index in the blockchain
tx_list	0...64000	An array list consisting all the transactions in the block
vdf_value	32	Value of the successfully computed mathematical Verifiable Delay Function
signature	64	Signature of a signed block

Table 5: Block structure

Each transaction contains the following main fields regardless of the transaction type:

Field Name	Size (bytes)	Description
tx_type	2	Type of the transaction (financial, signer_voting, producer_voting, etc)
from	4-32	Source account address who initiates transaction
vdf_value	32	Value of the successfully computed mathematical Verifiable Delay Function
signature	64	Signature of a signed transaction

Table 6: Main fields of each transaction

The structure of a typical financial transaction:

Field Name	Size (bytes)	Description
tx_type	2	Type of the transaction (financial, signer_voting, producer_voting, etc)
from	4-32	Source account address who initiates transaction
to	4-32	Destination account address who receives transaction
token	8	Amount of funds to send
fee	8	The amount of fee payment for a miner's transaction processing
vdf_value	32	Value of the successfully computed mathematical Verifiable Delay Function
signature	64	Signature of a signed transaction

Table 7: Financial transaction structure

Ansiblock will use variable size field 4-32 bytes for source and destination address mapping. Initially in Ansiblock transaction *from* and to fields are used to store full account addresses, addresses contain public keys of sender and receiver.

Public key length in Ansiblock network is 32 bytes, for optimizing transaction throughput Ansiblock proposes to use account indexing. Account indexing may help to reduce to and from field size down to 4 bytes. Account indexing may be used for majority of the financial transactions. 122 Bytes is the Ansiblock network minimal transaction size

2.4. Verifiable Delayed Functions

Ansiblock is using VDF values to guarantee fast and secure consensus of the network. A verifiable delay function (VDF) is a mathematical function that takes a predetermined amount of time to compute, even on a parallel computer. Output of VDF computed by one blockchain node can be quickly verified by others.

2.4.1. VDF Algorithms

The VDF function implemented in Ansiblock consists of three algorithms: **Setup**, **Eval** and **Verify**.

1. $Setup(\lambda, T) \rightarrow pp$ is a randomized algorithm that takes a security parameter λ and a time bound T, and outputs public parameter pp.

2. $Eval(pp, x) \rightarrow (y, \pi)$ takes an input x and output a y and a proof π.

3. $Verify(pp, x, y, \pi) \rightarrow \{accept, reject\}$ outputs $accept$ if y is the correct evaluation of the VDF on input x.

For complete definition VDF should have these three properties:

- ε **evaluation time** - Algorithm $Eval$ runs in time at most $(1+\varepsilon)T$
- **Sequentiality** - a parallel algorithm A using at most $poly(\lambda)$ processors, that runs in less than T cannot compute a function. Specifically, $Pr[A(pp, x) = y]$ is negligible.
- **Uniqueness** - for every input x exactly one y will be accepted by $Verify$

2.4.2. VDF Schemes

There are three popular ways to implement VDF (as of now):

- Trivial Scheme
- Pietrzak Scheme
- Wesolowski Scheme

Trivial Scheme

A trivial scheme for delay function can be built by iterating a cryptographic hash function. It can be designed as delay function which can be verified quickly. VDF can be constructed using incrementally verifiable computation(IVC) where proof of correctness can be computed in parallel.

Pietrzak and Wesolowski Schemes

Pietrzak and Wesolowski proposed two different VDF schemes. Both operates as follows:

- $Setup(\lambda, T)$ outputs (G, H, T), where G is finite abelian group, H is a efficient hash function $H : X \rightarrow G$.
- $Eval(pp, x)$ is defined as follows:
    - Compute $y \leftarrow H(x)^{2^T} \in G$ by computing T squarings in G starting with $H(x)$
    - Compute proof π
    - Output (y, π)

$Verify(pp, x, y, \pi)$ is where Pietrzak and Wesolowski proposals differ. Proof system of Wesolowski produces shorter proofs and verification is faster. However, the proof of Pietrzak has two advantages:

1. **Prover efficiency.** Pietrzak's prover takes $O(\sqrt{T})$ group operations to construct the proof, where as for Wesolowski it takes $O(T)$
2. **Comparison of the assumptions.** If Wesolowski's protocol is secure then so is Pietrzak's, but the converse is not known to be true.

Ansiblock VDF implementation is based on Pietrzak's scheme.

2.4.3. VDF Application in Ansiblock

Computational timestamping

Ansiblock is using Proof-Of-Stake algorithm for consensus. In Proof-Of-Stake we assume that supermajority of stakeholders participating in consensus are honest. But as soon as some of them try to intent maliciously being dishonest participants, they can conspire and may create an alternate transaction ledger. Currently blockchains prevent this through an external timestamping and prove that genuine history is older.

Ansiblock uses Incremental VDFs and their ability to provide computational evidence of passage of time. Ansiblock can prove that a given blockchain is older by proving that a genuine history has longer VDF computation.

Data Replication

Ansiblock provides data storage functionality over its blockchain. Ansiblock data storage layer is using VDFs to ensure that a number of replicas of the same file are being stored or given node is storing unique replica of some data.

Ansiblock node can store file by breaking it into b-bit blocks $B_1, ..., B_n$ and storing $y_1, ..., y_n$, where $(y_i, \perp) = Eval(pp, B_i \oplus H(id||i))$. To prove that node has a file a verifier can ask for a y_i. The result must be returned in significantly less time that one can compute Eval. The verifier can quickly verify returned value. If y_i is not stored, the VDF ensures that it cannot be computed quickly enough.

2.4.4. VDF Security Concerns

Implementing VDF in Ansiblcock has a concern if attacker creates ASIC for a VDF which will be N times faster (N=10-20) then standard CPU/GPU implementations attacker might be able to carry out a successful attack. In this case the adversary can compute the VDF so quickly that attacker can wrongly prove that alternate history is older than a genuine one. VDFs are also insecure against an attacker with a quantum computer.

2.5. Account Structure

Ansiblock has its own account structure supporting issuing and acquiring operations of various digital assets under single account. Ansiblock account structure was designed considering use cases for:

- Private Entities
- Corporate Entities
- Smart Contracts
- IoT Devices

Ansiblock account supports multiple signatures per transaction to provide additional security for Corporate Clients and operations with multiple parties involved.

2.6. Network Nodes and Clients

2.6.1. Clients

By developing API we support variety of clients to initiate/perform a transactions on the Ansiblock network:

- Mobile Wallets
- Web Wallets
- Desktop Wallets
- IoT Devices
- Applications

Ansiblock Wallet applications (supporting multisignature features), and Ansiblock Wallet API references source code (in multiple programming languages) will be published on GitHub for public access.

2.6.2. Nodes

Ansiblock introduces new node structure to provide to security and sub-second finality for transactions. In Ansiblock network nodes can play a several roles:

- **Producer**
  Producer is role of network node which is responsible for Block creation. Producer role is a temporary role for the node. Producers are elected through the election process on Ansiblock blockchain.

- **Signer**
  Signer is an important role of network node which is responsible for significant number of operations. Signers are processing blocks, validating broadcasted blocks, transporting transactions to signers, voting for blocks, participating in Producer elections, etc.

- **Collector**
  Collector is a role of network node which is responsible block replication and distributed data storage through Ansiblock Storage Layer. Network nodes can play Signer and Collector roles at the same time.

2.7. Layered Approach

Initialization Layer
Producers are forming Initialization Layer, they receive transactions from Ansiblock blockchain clients and initiate block creation process. Ansiblock is using Proof-of-Order mechanism (discussed in chapter 2.10.1 in more details) to ensure blocks are appended to distributed ledger properly and in secured way.

To achieve high performance, Producers residing on Initialization Layer are generating constant flow of transactions:

Typical Block Size	Typical Block Capacity	Typical Block Creation Frequency
64 kB	300-600 Transactions	1-5 Milliseconds

Table 8: Specs of the transactions flow

Ansiblock network throughput (TPS) depends on node interconnection speed:

Minimal Transaction Size (bits)	Node Interconnection Speed	Throughput (transactions per second)
976	1 gbps	1024590
976	500 mbps	512295
976	200 mbps	204918
976	100 mbps	102459

Table 9: Specs of TPS with different internet connection bandwidth

Initiation layer is broadcasting created blocks to the list of random nodes from Ansiblock Processing Layer.

Processing Layer

Ansiblock Processing Layer is most important layer of the our blockchain. Ansiblock Signer nodes are building blocks of Processing Layer which is responsible for significant part of intra-blockchain communications and Smart Contract executions. Processing Layer provides major inputs to Consensus Layer and contributes to low latency and finality of Ansiblock blockchain.

Storage Layer

Ansiblock Storage Layer provides distributed storage interface for whole Ansiblock blockchain data. Ansiblock Collector nodes are creating Storage Layer and provide inputs for Proof-of-State (PoST) mechanism (described in chapter 4).

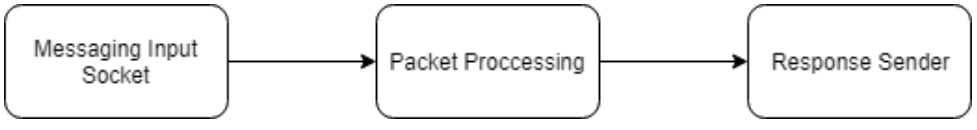
2.8. Ansiblock Communication Pipelines

Ansiblock network introduces several pipelines for different types of communications:

- Messaging Pipeline
- BlockGeneration Pipeline
- Broadcast Pipeline
- Transport Pipeline
- Synchronization Pipeline
- Voting Pipeline
- Reconstruction Pipeline
- Replication Pipeline

2.8.1. Messaging pipeline

Messaging pipeline is general purpose communication channel. Messaging pipeline is responsible for transporting messages from Ansiblock clients to network nodes. Inter-node communication is also handled by messaging pipeline. Ansiblock clients and other network nodes can query Signer nodes using Messaging Pipeline.



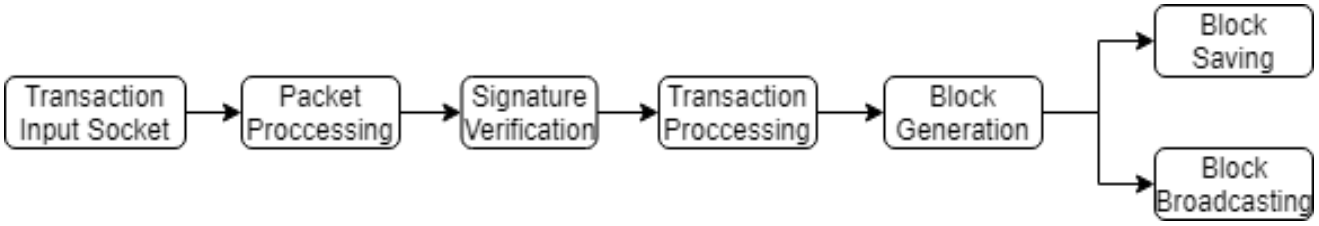
Pic.2 Messaging Pipeline Flow Diagram

Pseudocode of the Messaging Pipeline algorithm

```
while {
   packets := network.ReceivePackets()
   for each packet {
      message := packet.Deserialize()
      response := message.generateResponse(books)
      socket.send(packet.Address, response.Serialize())
   }
}
```

2.8.2. BlockGeneration Pipeline

BlockGeneration Pipeline plays significant role in Ansiblock network. BlockGeneration Pipeline is responsible for receiving Ansiblock client transactions, generating blocks and block broadcasting to Ansiblock network nodes.



Pic.3 BlockGeneration Pipeline Flow Diagram

BlockGeneration Pipeline can be initiated only on Producer nodes. Transaction Input Socket in BlockGeneration Pipeline is a dedicated UDP socket running on the Producer node. Ansiblock clients use Transaction Input Socket to initiate their transactions.

Packet Processing in BlockGeneration Pipeline is responsible for retrieving packets from a socket in the fastest possible way. Signature Verification process verifies that received packets are constructed and signed properly. Linux Network Stack is capable to receiving over 1 million packets per second, but Signature Verification cannot process packets at this rate and creates a bottleneck in the BlockGeneration pipeline.

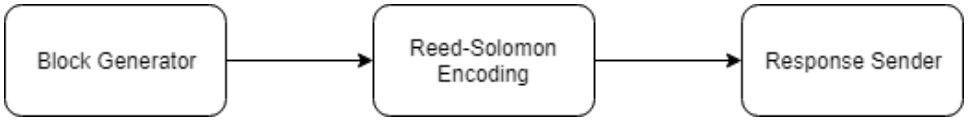
Transaction Processing in BlockGeneration Pipeline converts verified packets into transactions. For every transaction Transaction Processing verifies validity and checks if VDF Value is up to date and processes. Block Generation is creatings blocks for processed transactions. Blocks later are saved locally (Block Saving) on the node and broadcasted to the Ansiblock network (Block Broadcasting).

Pseudocode of the BlockGeneration Pipeline algorithm

```
while {
        packets := network.ReceivePackets()
        filteredPackets := books.SignatureVerification(packets)
        transactions := books.TransactionGenerator(filteredPackets)
        blocks := block.Generator(transactions)
        Broadcast(blocks)
}
```

2.8.3. Broadcast Pipeline

Broadcast Pipeline is used on Ansiblock network by Producer nodes for transfering created blocks to Signers nodes. The Producer node broadcasts each individual block to a random Signer node. When a Signer nodes receives a block, it shares a block to other Signer nodes using Transport pipeline. Before sending the blocks to the socket Signer node encodes blocks using Reed-Solomon algorithm, this ensures block data integrity and minimises the possible block data loss.



Pic. 4 Broadcast Pipeline Flow Diagram

Pseudocode of the Broadcast Pipeline algorithm

```
while {
        nodes := sync.NodesToBroadcast()
        encodedBlocks := erasure.encode(blocks)
        for i := encodedBlocks {
                send(encodedBlocks[i], nodes[i%len(nodes)])
        }
}
```

2.8.4. Transport Pipeline

Transport Pipeline on Ansiblock network is responsible for transporting blocks to Signer nodes on the network. To understand Transport Pipeline algorithm better imagine nodes as a tree structure. Where parent node is responsible for sending a block to the child nodes. Depending on number of child nodes of each node, Ansiblock can manage to maintain the constant depth of the tree.

Ansiblock introduces the idea of child and parent nodes, where every parent node may have up to 1000 child nodes, by using only 3 levels Ansibock can manage up to 1 billion nodes within this structure. Under this structure a block can be delivered to every node residing on the Ansiblock network in only 3 hops. Small number of hops dramatically affects the finality time. Ansiblock network nodes are “parenting” their nearest “child” nodes for algorithmic effectiveness.

Pic.5 Transport Pipeline Flow Diagram

Pseudocode of the Transport Pipeline algorithm

```
maxNumberOfHops = 3
while {
        blocks := network.ReceiveBlocks()
        for i := blocks {
                if blocks[i].hops >= maxNumberOfHops {
                        continue
                } else {
                        nodes := sync.NodesToTransit(blocks[i].hops)
                        for j := nodes {
                                send(blocks[i], nodes[j])
                        }
                }
         }
}
```

2.8.5. Synchronization Pipeline

Synchronization pipeline is communication channel on Ansiblock network which is responsible for the node data synchronization, detection of the new nodes or discovery of offline/dead nodes on the network. Node information as well as relevant socket addresses, identity of the Producer and latest valid VDF value are stored in NodeData structure.

Ansiblock uses state-based conflict-free replicated data type to synchronize data across nodes. Full local state is sent to other nodes, where it is merged by commutative, associative and idempotent function. Merge function in Synchronization Pipeline provides join for any two pair of states, to form a semilattice from set of states.

Update function in Synchronization Pipeline monotonically increases the internal state, according to the same partial order rules as the semilattice. Ansiblock achieves strong eventual consistency.

2.8.6. Voting Pipeline

Signer nodes are using Voting pipeline on Ansiblock network to vote on the Block with the State hash transaction. Every signer node verifies the State hash and sends a vote to the Producer and to other signers. When Producer node receives

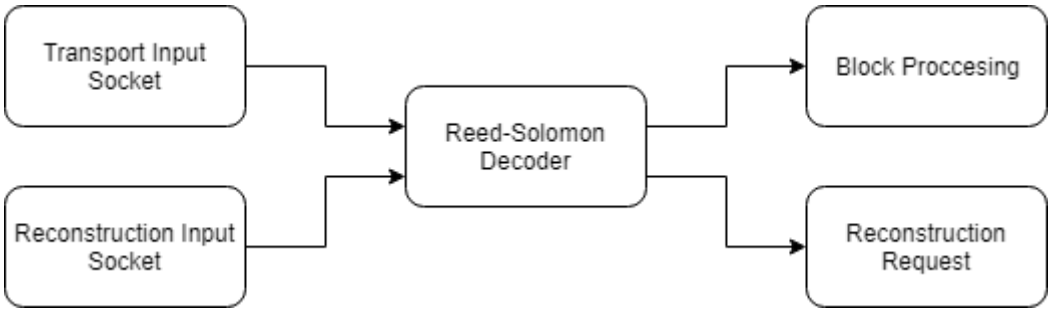
enough number of votes from Signer nodes, finality of the transaction is achieved on the Ansiblock network and next State Block is generated by Producer.

2.8.7. Reconstruction Pipeline

Ansiblock network uses UDP protocol for data transfer. UDP is connectionless protocol were packet losses may happen during data transmission. When Ansiblock Signer nodes discovers packet loss the missing data is queried from other Voting nodes through reconstruction pipeline.

Ansiblock nodes are performing following steps to eliminate packet loss and ensure data integrity:

1. Reed-Solomon algorithm implementation is used for lost data recovery.
2. If Reed-Solomon algorithm fails to recover data, (it may happen in very rare cases),Signer node will query missing data from other nodes using reconstruction pipeline.



Pic.6 Reconstruction Pipeline Flow Diagram

2.8.8. Replication Pipeline

Collector nodes are storing client transaction data on Ansiblock network. Replication Pipeline is used by Collector nodes for replicating data between each other. Replication pipeline makes sure the data is distributed properly over Collector nodes residing on Ansiblock network.

Collector nodes are depositing a collateral value to the Producer nodes before processing their transaction. Replication pipeline ensures that Collector nodes are responding to the challenges sent by other Ansiblock network nodes. Collector nodes failing to properly respond to the challenges are considered as malfunctioning nodes and their collateral is slashed.

2.9. Smart Contracts

Smart contracts are generalized transactions, which contain executable code as instruction. Ansiblock executes smart contracts on Processing Layer through Signer network nodes.

Ansiblock Smart Contracts implementation solves following challenges:

- Contracts are written by untrusted users, they should be sandboxed to ensure security.
- Contracts must executed in finite amount of time.
  (Infinite/long execution time can cause Denial-Of-Service for the Ansiblock network and influence transaction finality)
- Smart Contract development and SDK should be available for popular development languages.
- Parallel execution of Smart Contract for achieving high performance of the Processing Layer is required.
- Smart Contract versioning and change management.

2.9.1. Smart Contract Types and Versioning

Several types of Smart Contract are implemented in Ansiblock Processing Layer. We can separate them into two groups:

- Generic smart contracts
  (Smart Contracts designed to serve and maintain internal Ansiblock transaction processing)

- User developed Smart Contracts

Generic smart contracts are used to processing the following:

- Smart Contract for Enigma Private Transactions (which is described in section 3.3 below in more details)
- Smart Contract for Digital Asset Registration

Ansiblock supports smart contracts publishing and versioning functionality, by providing user interfaces and API for smart deployment and tracking we make this process flexible and transparent for smart contract developers and involved parties.

2.9.2. Ansiblock Virtual Machine

Aniblock strives to provide great flexibility and ease-of-use to Smart Contract developers. Generic steps for Smart Contract execution on varios blockchain networks are:

1. Development of Smart Contract
2. Smart Contract compilation to Virtual Machine bytecode
3. Bytecode execution on Virtual Machine

Ansiblock will support most popular programming languages for Smart Contract development (C, C++, C#, Crystal, Java, Javascript, Objective-C, Python, Ruby, Rust, Scala, Swift). Ansiblock implements support for multiple Virtual Machines, on initial stage we will support:

- eBPF (Extended Berkeley Packet Filter)
- WebAssembly

Running eBPF (Extended Berkeley Packet Filter) Virtual Machine

One of the Virtual Machines implemented in Ansiblick is eBPF (Extended Berkeley Packet Filter) virtual machine which was developed for Unix systems in 90's. In Linux kernel eBPF was further enhanced, more capability and performance were added.

Using eBPF as Virtual Machine provides several advantages to Ansiblock:

- eBPF is a kernel Virtual Machine, it is securely mature and provides strong sandboxing. Before executing the code eBPF bytecode is verified, if there is any illegal memory accesses or function calls.
- eBPF verify builds parse tree from bytecode and by DFS algorithm searches to every outcome of the program. If there are any loops or code does not come to the end, it will not be loaded. So every valid eBPF program comes to the end and there is no risk of malicious user will create infinite loop in the network. It can be reliably estimated how long program will run.
- The smart contracts can be written in supported programming languages that can be compiled to eBPF bytecode. The LLVM provides eBPF bytecode backend, so any frontend languages presented by the LLVM can be used by developers.
- Ansiblock examines memory requirements and preload necessary data required for each contract execution.
- Ansiblock will execute eBPF Virtual Machine on Signers' GPU. Contract execution (each contract) will happen in parallel on different GPU cores.
- Ansiblock performance proportionally depends on number of Signer nodes' GPU cores presented on the network.

Running Other Virtual Machines

Ansiblock implements support for multiple Virtual Machines, if they guarantee secure code execution and finality. The different Virtual Machine executions will run in parallel on Signer nodes on different GPU cores.

Together with eBPF (Extended Berkeley Packet Filter) Virtual Machine, Webassembly (WASM) Virtual Machine support will be implemented in Ansiblock Processing Layer. Ansiblock is not considering implementation of any other Virtual Machines at the moment.

2.10. Consensus Layer

Ansiblock is high speed blockchain network with low latency and finality. Consensus layer provides consensus for transactions on Ansiblock network based on four implemented proof algorithms:

- Proof-of-Order (PoO)
- Proof-of-Replication (PoRep)
- Proof-of-State (PoST)
- Proof-of-Stake (PoS)

2.10.1. Proof-of-Order (PoO)

Ansiblock uses Proof-of-Order (PoO) for computational timestamping. It is implemented using VDF as described above, to provide proof that incremental computation has been performed.

Proof-of-Order (PoO) Description

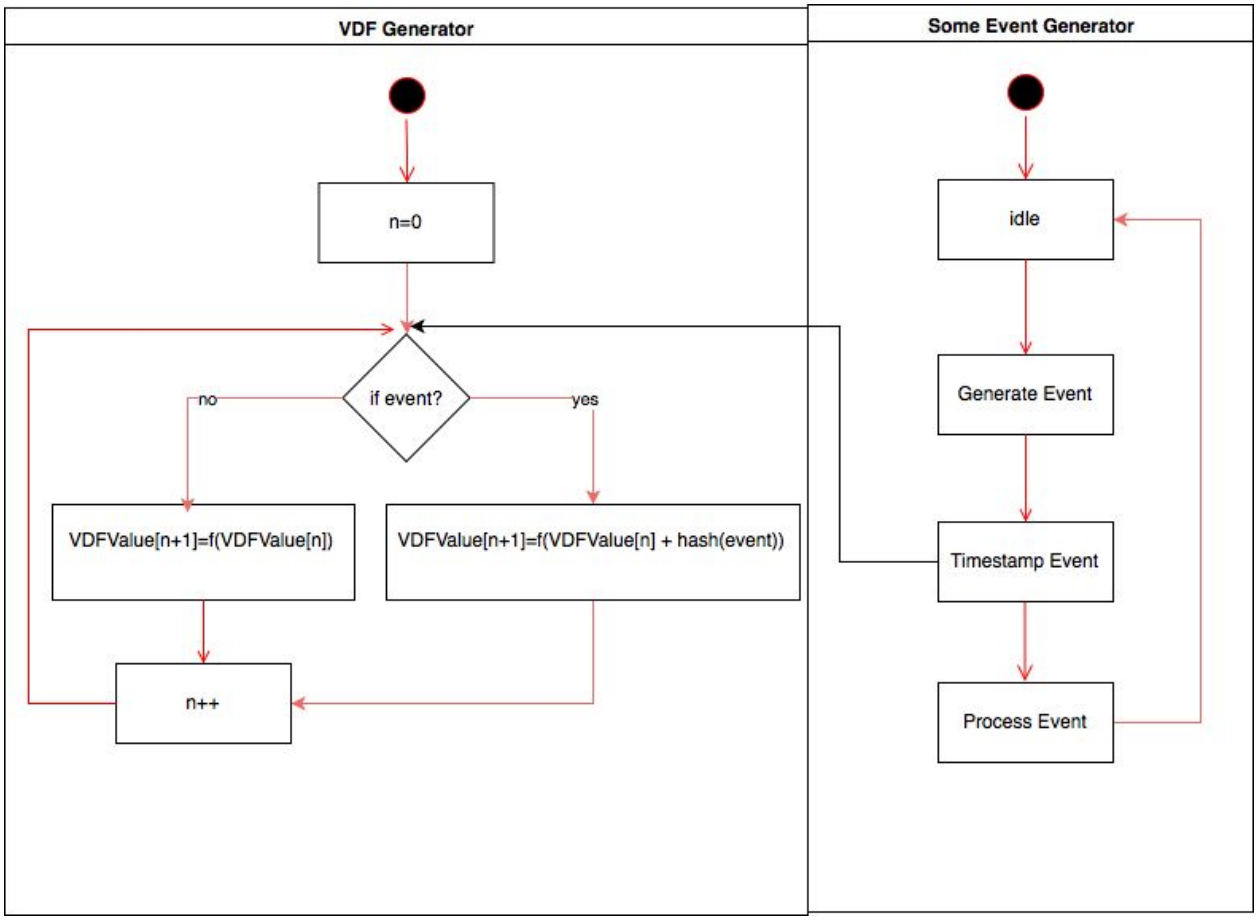
Proof of Order generator is run only on the Producer node. Producer constantly generates VDF Values using formula:

$VDFValue_n = f(VDFValue_{n-1})$,

Where f is a verifiable delay function.

Ansiblock might use VDF Values as some sort of computational timestamping as described in section 2.4. If it is required to timestamp some event Ansiblock should incorporate event hash into VDF chain using formula:
$VDFValue_n = f(VDFValue_{n-1} \oplus Hash(event))$

Constantly generated VDF Values determine the order of incorporated events. One can not say exact time event happened, but one clearly prove that event with $VDFValue_x$ happened earlier than an event with $VDFValue_y$ only if $x < y$. This is what we call Proof or Order.



Pic.7 VDF Generator Flow Diagram

Application of Proof-of-Order

Ansiblock network is timestamping every event in the blockchain using Proof-of-Order, forging blocks, sending transactions, votes, etc. Proof-of-Order is ordering blocks and transactions, using computed VDF Value, this is the way how Ansiblock verifies the order of the blocks later. Benefit of this process is that generation of VDF chain is serial and there is no theoretical possibility that someone can perform VDF computations in parallel. Verification process can be run significantly quicker compared to next VDF value computation.

It is vital for Ansiblock network to ensure consistency of VDF chain generation. Every event generated by blockchain client or Signer node carries last known VDF value. If Ansiblock client wants to make a transaction, last known VDF value should be included in the transaction data.

This is the way how Ansiblock clients are timestamping the transaction. Without timestamping malicious Producer has a possibility to reorder transactions. VDF values sent by Ansiblock clients are checked by Producer nodes. Transaction is processed only if Ansiblock Client's VDF value is in the chain of VDF values generated by the Ansiblock VDF Generator.

2.10.2. Proof-of-Replication (PoRep)

Ansiblock network stores client transaction data in decentralised way by implementing interactive Proof-of-Replication (PoRep) algorithm, where it can be verified that prover is storing a replicated data. Proof-of-Replication (PoRep) was initially introduced by Filecoin project.

Definition of PoRep

A PoRep handles any data $D \in \{0,1\}^*$, as follows:

1. $PoRep.Setup(\lambda, T) \rightarrow pp$ is a randomized algorithm that takes a security parameter λ and a time parameter T, and outputs public parameter pp.
2. $PoRep.PreProc(D) \rightarrow D', \tau$ is a preprocessing algorithm that takes data D and outputs preprocessed data D' with its tag τ
3. $PoRep.Replicate(id, \tau, D') \rightarrow R, aux$ takes replica identifier id and preprocessed data and outputs its replica R and a compact auxiliary information to pass to $Prove$ and $Verify$ procedures.
4. $PoRep.Extract(pp, id, \tau, R) \rightarrow D'$ outputs data D' and verifies its consistency with tag τ.
5. $PoRep.Prove(R, aux, id, r) \rightarrow \pi$ outputs a proof π_{id}
6. $PoRep.Poll(aux, B) \rightarrow r$ outputs public challenge r, derived from random beacon B.
7. $PoRep.Verify(id, \tau, r, aux, \pi) \rightarrow \{accept, reject\}$ accepts or rejects the proof.

Verifiable Delay Encodings

A verifiable delay encoding (VDE) is a special case of a verifiable delay function(VDF) discussed in section 4. Similar to VDF, it's a slow to compute encoding that is fast to decode. The encoding requires non-parallelizable sequential work and cannot be computed in shorter time.

VDE algorithms are working as described below:

1. $VDE.Setup(\lambda, T) \rightarrow pp$ is given security parameter λ and delay parameter T produce public parameters pp.
2. $VDE.Enc(x) \rightarrow y$
3. $VDE.Dec(pp, y) \rightarrow x$

Proof-of-Replication (PoRep) with VDE

Ansiblock implementation of Proof-of-Replication (PoRep) uses VDE to get a unique replica R_{id} with slow encoding. The Verifier can check that Collector is storing R_{id}. If Collector will delete replica R_{id} node will not be able to re-derive it in time. One should not be able to get the replica for some id_1 with the replica of id_2. For this purpose we will be using a random oracle H applied to the replica id and decode $D \oplus H(id)$ instead of D. Recovery of D is still possible as id is known to the decoder. To be more specific we will break the data up to some N chunks and encode each $D \oplus H(i||id)$.

Let $O_{check}(d_i, i, \tau) \rightarrow b \in \{accept, reject\}$ be the verification method for the chunk d_i. The algorithms are as follows:

1. $PoRep.Setup(\lambda, T) \rightarrow pp$. This specifies the chunk length m and provides input parameters to $VDE.Dec$ and $VDE.Enc$. Also specifies $l = l(\lambda)$

2. $PoRep.Replicate(id, \tau, D') \rightarrow R, aux$. Parse D' as N chunks $d_1, ..., d_N \in \{0,1\}^m$. Compute $R_i = VDE.Enc(d_i \oplus H(id||\tau||i)$. Output $R = (R_1, ..., R_N)$.
3. $PoRep.Extract(pp, id, \tau, R) \rightarrow D'$. Parse $R = (R_1, ..., R_N)$ and compute $d_i = VDE.Dec(R_i) \oplus H(id||\tau||i)$. Output $D' = (d_1, ..., d_N)$
4. $PoRep.Poll(N) \rightarrow r$. For $i = 1$ to l randomly sample $r_i \leftarrow [N]$. Output $r = (r_1, ..., r_l)$
5. $PoRep.Prove(R, N, id, r) \rightarrow \pi$. Output proof $\pi = (R_{r_1}, ..., R_{r_l})$.
6. $PoRep.Verify(id, \tau, r, aux, \pi) \rightarrow \{accept, reject\}$. Parse proof $\pi = (\pi_1, ..., \pi_l)$ as l strings in $\{0,1\}^m$.

Pseudocode of the VDE verification algorithm

For each $i = 1$ to l do {
Compute $d'_i = VDE.Dec(\pi_i) \oplus H(id||r_i)$
Query $b_i \leftarrow O_{check}(d'_i, r_i, \tau)$

}
If $b_i = 1$ for all i then output $accept$, else output $reject$

Proof-of-Replication (PoRep) Application in Ansiblock

Producer and Collector nodes on Ansiblock network are participating in Proof-of-Replication (PoRep). Block created by Producer nodes are broadcasted to Collector nodes. Collector nodes are forming Storage Layer in Ansiblock network and provide data storage and replication functionality.

Collector nodes are becoming effective and join Storage Layer after they pledge their storage and deposit collateral to the Ansiblock network. Collector nodes are receiving broadcasted blocks on the network and respond with the commitment to store the data. Later Collector nodes are generating Proof-of-Replication (PoRep) and summit it to the Ansiblock Storage Layer to prove that data is stored. Ansiblock validates Collector node commitments, Collectors are penalised for invalid or missing proofs, and may lose their collateral for malfunctioning.

Producer nodes are polling Collector nodes for the Proof-of-Replication (PoRep). Periodically Producer nodes are sending a challenge to the Collector nodes. Challenge is generated randomly using latest VDF value available in the VDF Generator. Collector nodes output the Proof-of-Replication (PoRep), which is validated by other nodes residing on the Ansiblock Storage Layer. Proof-of-Replication (PoRep) is considered verified when super majority of Collectors nodes confirm it. The verifications are collected using Replication Pipeline.

2.10.3. Proof-of-State (PoST)

Proof-of-State (PoST) is cryptographic proof that blockchain is in specific state. State of the blockchain is a “snapshot” of Ansiblock network containing all client data, including accounts, smart contract and other data entities of the blockchain.

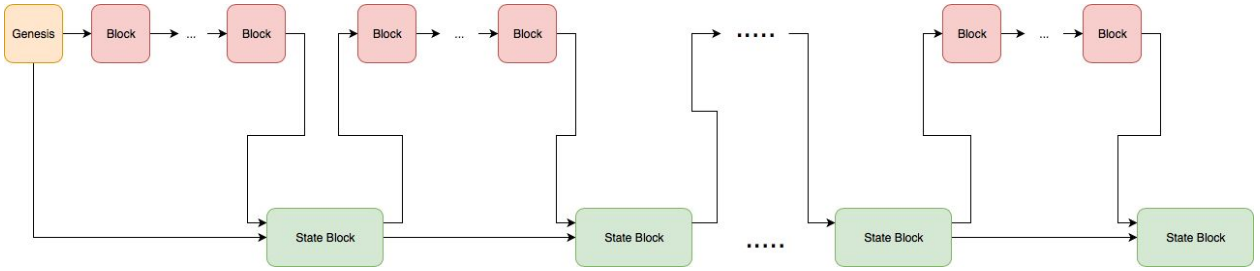
Challenges of a high performance

Ansiblock network performance may reach 1,024,000 transactions per second (with 1gbps internet connection bandwidth), what potentially may generate blockchain data for Storage Layer up to 100 Megabytes per second, 8 Terabytes per day, 3 Petabytes per year.

New Collector nodes willing to join Ansiblock network will inevitably experience data synchronization difficulties. Process of initial data download from the Ansiblock network will be extremely time consuming and costly. Ansiblock network introduces Proof-of-State (PoST) to save and verify the state of whole blockchain. Proof-of-State (PoST) will allow new Collector nodes on Ansiblock network to download and verify the final state of the blockchain and start processing of upcoming block.

Proof-of-State (PoST) Implementation

Ansiblock network introduces State Block and Hash-of-State to manage Proof-of-State (PoST)process. State block is a single transaction block, which contains current Hash-of-State, current VDF value and Signer nodes signatures.
Hash-of-State is implemented thought SHA256 of current data in Storage Layer of the Collector node.



Pic. 8 Proof-of-State (PoST) Flow Diagram

State blocks are been generated periodically on Ansiblock network in every f seconds. Signer nodes are obligated to verify the hash of the block and place a vote for this block. New Collector nodes joining Ansiblock Network are downloading chain of the State Blocks and Final State.

Nodes will be able to confirm that:

1. Final state on Ansiblock network is the result of the genesys state.
2. Time has passed during the process (implemented through of VDF) and states are ordered correctly (implemented through of Proof-of-Order (POO))
3. Signers nodes really voted for all consequent states.

Ansiblock nodes do not know exactly how states were changed (list of transactions), but they can confirm that state downloaded is the correct one.

2.10.4. Proof-of-Stake (PoS)

Proof-of-Stake (PoS) general implementation in Ansiblock network relies on the list of Signer nodes which agree on new block creating by reaching consensus with each other.

Producer nodes or new nodes joining Ansiblock network can become Signers by sending a special collateral transaction. Ansiblock network is using collateral transaction to lock up desired amount of tokens as a node deposit.
Signers nodes are randomly assigned Producer node role on Ansiblock network, with the right to propose blocks. Signer nodes agree on proposed blocks through a voting process. Every Signer node sends a “vote”, the weight of the “vote” is proportional to the Signer’s deposit. When the supermajority of the Signers(⅔ of the stake) agree on a block, then block is considered forged and finality is achieved.

Proof-of-Stake (PoS) in the Ansiblock

Ansiblock network is using Byzantine Fault Tolerant style (partially synchronous) Proof-of-Stake (POS) algorithm. Trough making a deposit on the Processing Layer node may become Signer node. Deposits are made via special transaction that is incorporated into the block by Producer node and later processed by other Signer nodes.

After becoming a Signer node is required to verify State blocks and submit the vote. They must submit their votes within some amount of time. VDF chain enables us to timetrack the votes. Every vote is just a transaction in the network viewed by all the nodes. It is important to verify that vote was submitted before observing other votes and whole voting process. Ansiblock network can verify that using computational timestamping. The power of the vote depends on the deposit of the node. The Ansiblock uses slashing for solving the nothing at stake problem.

Proof-of-Stake (PoS) Workflow

Producer node receives transactions from users and creates new block (VDF value is included in the block for Proof-of-Order). Producer node broadcasts block to the Ansiblock network. Ansiblock network verifies the correctness of the block. New Producer node is elected on Ansiblock network and old Producer node is slashed, if block is not correct.

After time f Producer node sends state hash into the next block as a special transaction. This must be verified by super majority of the Delegate Signers. Delegate Signers send vote transactions to the Producer node and to each other. This way Signers verify that state they are in is the same and it is the valid state. Ansiblock network is reaching finality in time f . If the state is not verified (every Signer node detects this independently) than Producer node is slashed and new Producer node is elected. Verified state block (with Signer verifications) is appended to the blockchain.

Every time Ansiblock network has N Delegated Signers to vote for the state block. State block is generated in every f time. Ansiblock network will have N vote transactions in f time. The number we get is significantly small compared to all the other transactions in the network. f and N are parameters that could be modified, but we assume $f < 1\ second$ and $N < 10000$. In this case Ansiblock network can process 1,024,000 transactions in a second, only 10,000 of these transactions are votes.

Slashing conditions

Ansiblock defines rules that determine when a given Signer node can be deemed beyond reasonable doubt to have misbehaved (e.g. voting for multiple conflicting blocks at the same time). If a Signer node triggers one of these rules, their entire deposit gets deleted.

Producer nodes are slashed in following cases on Ansiblock network:

1. Producer node is censorings some Signer node;
2. Signer nodes detect that Producer node generates more than one block at the same time;
3. Signer nodes detect that Producer node generates incorrect block or the Hash-of-State.

The Ansiblock network has following slashing conditions for Signer nodes:

1. Signer votes on two different sequence of blocks.
2. Signer votes on the invalid VDF value generated by Producer
3. Signer is down for extended period of time.

Finality

Finality conditions are rules that determine when a given block is considered finalized. If ⅔ of the Signer nodes voted for the block than it can be economically finalized. Ansiblock can prove that if some other block is also finalized honest Signers will not be punished and byzantine Signers lose their stake.

To achieve safety ⅔ of the Signers should be honest as is stated in traditional byzantine fault tolerance theory. In case of safety failure at least ⅓ of Signers are misbehaving and Ansiblock network can detect who are they, prove it and slash them.

Elections

Election is a process of choosing a role of Producer node or a Delegated Signer node. Producer node is elected deterministically depending on Signer's proportion of stake. In long run each Signer node will play a role of Producer node proportional to their stake. Ansiblock network uses last verified state's VDF value to determine the Producer node. There is no possible way to predict which node will be Producer node in advance.

In case of partition the same node will be Producer node in each network. So deterministic election eliminates double spending attack.

Election triggers on Ansiblock network are:

1. Time passage - Ansiblock network can clearly detect passage of time using VDF chain
2. Incorrect VDF value generated by the Producer node
3. Incorrect block generated by the Producer node
4. A network timeout
5. Detected fork - Fork occurs only when Producer is compromised. That means two different sequence was broadcasted by same Producer.

Delegate Signer Elections

Delegate Signer nodes are elected deterministically on Ansiblock networks using the last verified state's VDF value and taking into consideration their stake size. Ansiblock network publicly provides the list of Delegate Signer nodes. Delegate Signer nodes have same voting power and they receive same award. Sum of the fees of the block is divided among Delegate Signer nodes.

To protect Signer nodes from Producer node's censorship each Signer node will send their vote to delegates and the Producer node. The delegates will converge recieved votes with each other. The Producer node will propose block with Signers rewards. If Producer node excluded some Signer, Delegates will be able to detect censorship as they get all the votes. In case of censorship Producer node is reelected and it's deposit is slashed.

Partition

In case of network partition each part knows who is the Producer node. If Producer node is in the part where supermajority of Signers are (by their stake), then block creation process continues as usual. If Producer is in the lesser half then Producer will not be able to verify the state block (supermajority is required). Ansiblock will start the forceful unstaking process, when Signers will lose their voting power. Unstaking speed will depend on the partition size so that bigger part in the partition will finish unstaking faster and will be able to generate and verify new state blocks.

2.11. Security Layer

Building a highly secure blockchain is a major challenge for decentralised technology providers and innovators, Ansiblock security layer provides protection against all attacks known to us:

- Double spending attacks
- Grinding attacks
- Transaction denial attacks
- Desynchronization attacks
- Eclipse attacks
- Long-range attacks
- Nothing at stake attacks
- Sybil Attacks
- Outsourcing Attacks and Generation Attacks

Double spending attacks
The malicious client generates two transactions in parallel for the same tokens with different recipients. The goal is to double spend the same money. Also the client could wait network to process first transaction and then send the second to revert payment. In first case neither Producer nor Signers will pass double spending in the same block. In the second case Proof-of-Order provides order of transactions, the second transaction will be rejected as the same token was processed earlier.

Grinding attacks
In Grinding attack malicious Signer tries to influence Producer election process to increase chances to become the Producer itself. Ansiblolck elects with deterministic algorithm, the Signer with largest stake becomes the Producer. Also supermajority of votes should confirm the new elector. If some Signer will double votes for different electors, it will be detected by others and it's stake will be slashed.

Transaction denial attacks
In a transaction denial attack, the goal of adversary is to prevent the target's transactions to be processed and committed by network. There are to cases:

1. The attacker is a Producer itself. It can censor any transaction as it chooses what transactions will be included in the block. The Ansiblock frequently elects the new Producer that will process blocked transactions. Eventually all transactions will be included in blockchain.
2. If more than ⅓ of Signers do not vote for certain target's transactions, there is no way to protect from such an attack, but Ansiblock can make it costly. As the network observes all votes, Ansiblock network can penalize Signers who did not participated in voting. This will make censorship expensive.

Desynchronization attacks
In a desynchronization attack, a Signer behaves honestly but is nevertheless incapable of synchronizing correctly with the rest of the network. This leads to ill-timed issuing votes. Desynchronization can occur because of network delays. The Ansiblock synchronization pipeline tracks liveness of the node. Each node has to vote in fixed period of time, the network penalizes nodes for not participating in the voting and also dynamically adjust how fast Signer's stakes become stale. If the node can not keep up, eventually it's stake will be slashed and it will lose voting power. No guarantees of liveness and persistence are provided for desynchronized parties.

Eclipse attacks
In an eclipse attack, message to the Signer or from the Signer are subverted. For the network this is the same case as desynchronization attack. The Signer can protect itself by reconstructed lost blocks form random nodes. Randomization will make the attack harder.

Long-range attacks
An attacker who wishes to double spend at later point in time can locally revert blockchain to genesis block. So it will be the only Producer/Signer. It can generate new blocks faster as no networking is involved. At the point when attackers chain is longer than main chain, it can spend tokens get the goods and then send block from faked chain to the network. The faked chain will be longer, so Signers should switch to it. The Ansiblock is protected from long-range attack by PoO, the attacker has to incorporate all the old transactions in it's chain, as VDF is not parallelizable, it has to process transactions on one core much faster than current network does.

Nothing at stake attacks
The "nothing at stake" problem refers the attack against PoS algorithm. Holding a voting stake does not require expensive computing power. In case of partition the node can vote in all partitions to maximize earning. In the Ansiblock eventually partitions will merge and the network will be able slash the stakes of malicious Signers.

Sybil Attacks
Malicious Collectors could pretend to store (and get paid for) more than one copy of data by creating multiple sybil identities. It will store only one copy of data, but provide information with multiple identities. To protect from such case, each copy of data will be encoded with different encoding. The malicious Collector will have to keep data decoded, in case of query it will encode data with different coding for different identities. The Ansiblock uses VDE, so encoding is harder and the Collector will not be able encode data in time, it's stake will be slashed and will lose status of the Collector.

Outsourcing Attacks and Generation Attacks
Malicious Collectors could commit to store more data than the amount they can physically store, relying on quickly fetching data from other storage providers. Malicious Collectors could claim to be storing a large amount of data which they are instead efficiently generating on-demand using a small program. To check Collectors, each of them gets different challenge C, each Collector has to produce proof independently. If the malicious Collector will need to retrieve data and encode it incorporating with challenge. Because of additional steps to produce proof, the process will be distinguishable slow, it will be detected by network and the Collector will be slashed.

3. Privacy Layer

Blockchain economy heavily relies on open ledger philosophy where transaction history happening in the network is open and available for all network participants. Placing client transactional data on open ledgers, is one of the major obstacles for financial institutions willing to adopt blockchain technologies. Financial institution are following regulatory and compliance requirement to protect privacy of their clients.

Idea of having private transaction processing over blockchain network not only contradicts to the concept of open ledgers running in trustless environment, but also introduces a big technological challenges for current blockchain technologies. Implementation of privacy layer for transactions processing requires extra resources from the network and cripples performance. Centralized or off-chain solutions for providing privacy layer, create additional issues with trust and availability.

Ansiblock offers implementation of Privacy Layer as a solution of institutions willing to process client transactions privately (without disclosing transaction details to other network participants) on Ansiblock blockchain.

Ansiblock Privacy Layer is implemented on top of our Ansiblock network and not affects its performance. Ansiblock is Privacy Layer is implemented through Enigma Contracts. Ansiblock Enigma Contracts are internal high performance smart contracts for private transaction processing.

3.1. Key Generation

Ansiblock sender can generate one-time key from random data and receiver's address. Ansiblock private transactions are using one-time key. Ansiblock sender account should validate private transactions destined for receiver. Only the receiver address on Ansiblock network will be able to process transaction and receive funds.

Proposed algorithm works as follows:

1. Receiver party on Ansiblock network generates two private public key-pair $A = a * G,\ B = b * G$, where group G elements are encoded as in $ED25519$.
2. Receiver party on Ansiblock network sends to Sender A, B.
3. Sender party on Ansiblock network generates random key-pair $R = r * G$
4. Sender party on Ansiblock network computes one-time public key $P = H(r * A \,||\, n) * G + B$. Where H is a collision resistant hash function and n is a nonce.
5. Sender party on Ansiblock network generates private transaction with P and R, includes amount he/she wants to send and signes the transaction.
6. Receiver party on Ansiblock network checks every private transaction to find the one that was sent to him/her.
7. Receiver party on Ansiblock network computes $\widehat{P} = H(a * R \,||\, n) * G + B$
8. If $\widehat{P} = P$ then transaction was sent to receiver.
9. Secret key p for P ($P = p * G$) is $p = H(a * R \,||\, n) + b$.

It is worth mentioning that for private transaction detection one has to have keys a, B and for secret key p one has to have keys a, b.

3.2. Linkable Ring Signatures

Ring signatures can verify some number of public signatures. So one can prove that he/she is a part of a group(ring) without revealing who is he/she. Linkable ring signatures in addition prove that signer has never produced a signature for the ring. They have these three algorithms:

1. $LRS.Sign(sk, R, m) \rightarrow \sigma$, where sk is a secret key of the signer, R is ring of public keys and m is the message.
2. $LRS.Verify(R, m, \sigma) \rightarrow \{accept, reject\}$ accept or reject the proof.
3. $LRS.Link(\sigma_1, \sigma_2) \rightarrow \{true, false\}$ determine if signatures σ_1 and σ_2 are produced by the same signer.

With the help of Linkable Ring Signatures we can ensure that anonymous signer can withdraw funds without double spending.

3.3. Enigma Contract

Ansiblock Enigma contract is a high performance smart contract running on network. Ansiblock Enigma contract is responsible for fast transferring of private transactions.

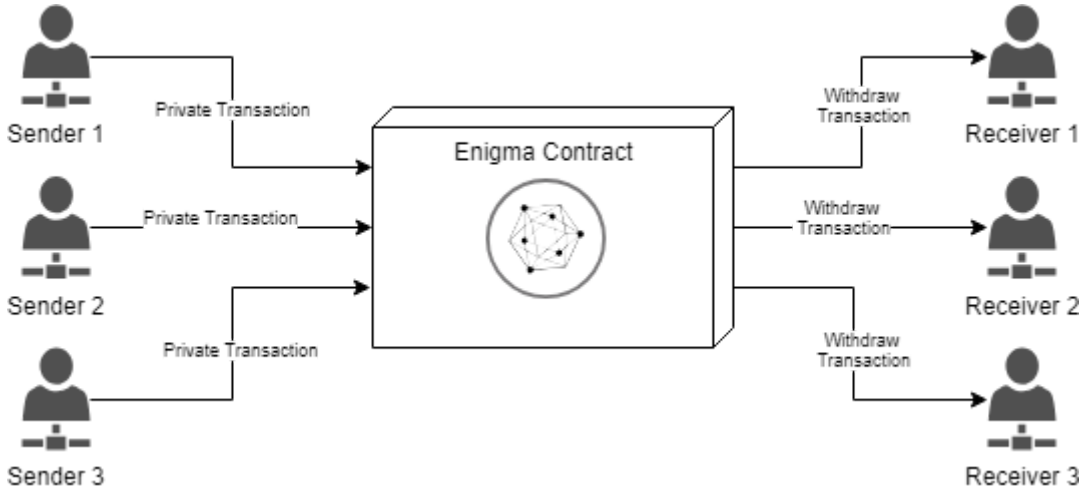
Ansiblock will generate Enigma Contracts to process private transaction To transfer private transactions of arbitrary amount, the Ansiblock will generate many Enigma Contracts, (for every power of two described later). Ansiblock also add Enigma Contracts for other common denominations for convenience.

Fund Transfer Principle via Ansiblock Enigma Contract

Let's assume everyone in the Ansiblock network is transferring amount which equals to 1 Ansiblock Token (AT). Enigma Contract gathers incoming private transactions in the ring and outputs funds to recipients.

Ansiblock network cannot (since it does not holds the secret key) determine the from-to pair, and all private transfers have amount which equals to 1 Ansiblock Token (AT).

Ansiblock clients can prove to the Enigma Contract that they are private transfer destination party for transfered amount of 1 Ansiblock Token (AT) token using Linkable Ring Signatures. After successful completion of verification process Enigma Contract transfers funds to destination party account with Withdraw Transaction.



Pic.9 Enigma Contract Flow Diagram

3.4. Private Pipeline Algorithm

Ansiblock Private Pipeline is processing client transaction in private fashion. Ansiblock Private Pipeline algorithm works as follows:

1. Ansiblock sender party wants to transfer X tokens to the Ansiblock receiver party
2. Ansiblock sender party breaks X to the powers of two (or better decomposition, managed by Ansiblock Enigma Contract)
3. Ansiblock sender party generates private transactions using one-time keys and sends the decomposed amounts to the corresponding Enigma Contracts.
4. Ansiblock Receiver party scans private transactions in Enigma Contracts, to find a transaction which destined for given Receiver account.
5. There are many other private transactions in Enigma Contract, ones that are destined for different Ansiblock Receivers. Ansiblock Receiver Party generates a Ring Signature with:

    - Receiver Party Private key
    - Public Keys for other parties
    - Account address were Receiver Party wishes to receive the funds

6. After generating Ring Signature Ansiblock Receiver Party creates a withdraw transaction and send it to Ansiblock Enigma Contract.
7. Ansiblock Enigma Contract receives a withdraw transaction, confirms the ring signature (using $LRS.Verify$), confirms uniqueness of the signature (using $LRS.Link$) and withdraws funds to the provided address.

Ansiblock Enigma Contracts are building undetectable fence for private transactions and eliminating the link between Sender and Receiver party in the blockchain network.

3.5. Private Pipeline for Corporate Entities

Ansiblock brings corporate entities possibility to use blockchain for processing their clients' transactions. By implementing Private Pipeline for Corporate Entities Ansiblock helps corporate clients to leverage blockchain technology without running and maintaining private blockchains inside their organisation. Corporate entity such as a bank, payment processor or governmental institutions can use Ansiblock Private Pipeline and Verifiable Private Transaction Contracts (VTP Contracts) instead of running a separate private blockchain..

Ansiblock Verifiable Private Transaction Contracts (VTP Contracts) and Ansiblock Enigma contracts are following the same principles. The privacy of transactions is achieved by same algorithms - private transactions and linkable ring signatures.

To use Private Pipeline for Corporate Entities, corporate clients need to define their own Verifiable Private Transaction Contracts (VPT Contract). In Verifiable Private Transactions sender should include private key a in the transaction, encoded with public key of corporate entity. This way corporate entity (and other holders of private key) will be able to

recover private key a of the sender. With private key a and public key B corporate entity will be able to detect transaction from sender to receiver. Without private key b corporate entity will not be able to transfer funds from smart contract to its account.

4. Interoperability Layer

Development of blockchains technologies emphasis the importance of cross-chain (communication between different blockchains). Ansiblock introduced Interoperability Layer as interconnection protocol enabling cross-chain communication and data flow between Ansiblock and other blockchains support cross-blockchains interactions.

Ansiblock Interoperability Layer is implemented as staged protocol where cross chain transaction are initiated and finalised. Transaction processed by Interoperability Layer are passing through following stages:

- **Initiation Stage**
  Ansiblock identifies each cross-chain transaction and encapsulates them into special structure to provide extra level of security and reliability.

- **Transmission Stage**
  Ansiblock network initiates connection with destination chain and starts cross-blockchain data exchange provides validation for exchanged data.

- **Finalisation Stage**
  Ansiblock Interoperability Layer manages cross-blockchain transaction finalisation on Finalisation Stage.

Ansiblock Interoperability Layer provides cross-blockchain transaction status tracking and monitoring interface.

8. References

[1] D. Boneh, B. Bunz, B. Fisch. A Survey of Two Verifiable Delay Functions
[2] J. Benet, D. Dalrymple, N. Greco. Proof of Replication. Protocol Labs, 2017
[3] J. Buchmann, S. Hamdy. A survey on iq cryptography. In Public-Key Cryptography and Computational Number Theory, 2001
[4] R. Rivest, A. Shamir, D. Wagner. Time-lock puzzles and timed-release crypto, 1996
[5] B. Cohen. Proofs of space and time. Blockchain Protocol Analysis and Security Engineering, 2017
[6] C. Dwork. N. Lynch, L. Stockmeyer. Consensus in the Presence of Partial Synchrony, 1988
[7] Protocol Labs. Filecoin: A Decentralized Storage Network, 2017
[8] C. Badertscher, P. Gazi, A. Kiayias, A. Russell, V. Zikas. Composable Proof-of-Stake Blockchains with Dynamic Availability, 2018
[9] M. Castro, B. Liskov. Practical byzantine fault tolerance, 1999
[10] S. King, S. Nadal. Peer-to-peer-currency with proof of stake, 2012
[11] V. Buterin, Minimal Slashing Conditions, 2017
[12] B. Fisch, PoReps: Proofs of Space on Useful Data, 2017
[13] B. Libert, S. Somindu, C. Ramanna, M. Yung. Functional commitment schemes: From polynomial commitments to pairing-based from simple assumptions, 2016
[14] P. Valiant. Incrementally verifiable computation of proofs of knowledge imply time/space efficiency, 2008
[15] B. Wesolowski. Efficient verifiable delay functions, 2018
[16] J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In H. Wang, J. Pieprzyk, and V. Varadharajan, editors, ACISP 04, volume 3108 of LNCS, pages 325–335, Sydney, NSW, Australia, July 13–15, 2004. Springer, Heidelberg, Germany.
[17] Steven McCanne, Van Jacobson, 1992:
http://www.tcpdump.org/papers/bpf-usenix93.pdf
[18] eBPF, Linux:
https://www.kernel.org/doc/Documentation/networking/filter.txt
[19] Benjamin Wesolowski. Slow-timed hash functions. Cryptology ePrint Archive, Report 2018/623, 2018:
https://eprint.iacr.org/2018/623
[20] Casper the Friendly Finality Gadget:
https://arxiv.org/pdf/1710.09437.pdf
[21] Ben Fisch, Joseph Bonneau, Juan Benet, and Nicola Greco. Proofs of replication using depth robust graphs. In Blockchain Protocol Analysis and Security Engineering 2018, 2018:
https://cyber.stanford.edu/bpase2018
[22] Ethereum Wiki, Proof-of-Stake FAQ, 2016: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs
[23] Filecoin, proof of replication:
https://filecoin.io/proof-of-replication.pdf
[24] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. Cryptology ePrint Archive, Report 2018/601, 2018:
https://eprint.iacr.org/2018/601
[25] Slasher, A punitive Proof of Stake algorithm:
https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm
[26] Filecoin: A decentralized storage network. Protocol Labs, 2017:
https://filecoin.io/filecoin.pdf
[27] K. Pietrzak. Simple verifiable delay functions. Cryptology ePrint Archive, Report 2018/627, 2018:
https://eprint.iacr.org/2018/627