Chandler Song, Stanley Wu, Ryan Fang
info@ankr.network

ANKR Network

A Resource-Efficient and Scalable Blockchain

Preliminary Draft. Private Information.   April 25

A N

K R

ANKR Network ANKR Network

A N

K R

Table of Contents

Abstract

In this white paper, we present the Ankr Blockchain, a multi-chain and useful work based decentralized network with a native data feed. We first review a novel mining solution using the **Proof of Useful Work (PoUW)** consensus. We will explain the **Multiple-Chain Structure** and then describe how we plan to provide an generic interface for **Town Crier**, an authenticated data feed for smart contracts. We will also discuss how Ankr's technologies will enable **decentralized cloud computing**. Finally, we will characterize the **Ankr Ecosystem** and the app interface to further facilitate the utilization of the Ankr network.

1 Introduction

1.1 Current Blockchain Technology

A blockchain is a growing list of blocks. These blocks contain records and timestamps. A blockchain network is a decentralized network, where every node on this network keeps a copy of the information on this network. In this way the network is immune to single point of failure. The network is decentralized where users need to only trust the consensus, not anyone else, to validate a piece of information. This solves Byzantine General Problem. These properties make blockchain an ideal vehicle for public ledger. A pioneer application of blockchain is Bitcoin.

Since the advent of Bitcoin, the world has witnessed rapid progress and improvement of blockchain technology. From Bitcoin, a P2P decentralized censorship-resistant electronic cash system that was primarily built as a transaction application, to Ethereum, the next major innovation with programmable languages that support customized applications through "Smart Contract" technology, to countless other endeavors of the blockchain community, the advantage and benefits of a decentralized framework have become more evident over time.

It is common for today's companies to try to integrate blockchain technology into their existing business model. However, there are still various obstacles preventing mass adoption of the blockchain technology in real-world business applications, and we believe three of the biggest challenges are

- Massive waste of power by current mainstream blockchain technology
- Lack of reliable and efficient oracle (data feed) services to connect existing businesses to the blockchain
- Low TPS and lack of scalability due to the overloaded single blockchain implementation for all business purposes

At this moment, there are two design paradigms for blockchain applications:

- Treating a blockchain as a database with all business logic residing on the applications – such frameworks will inevitably slow down over time as exponential data growth will bloat the blockchain size
- Collecting smart contracts for drastically different business use cases in one blockchain – such framework will face considerable difficulty to maintain and scale up in the long run

Most blockchain applications use either Proof of Work (PoW) or Proof of Stake (PoS) as the consensus for validating a piece of information (in most cases the information is the ownership of a coin or a token).Yet both Proof of Work and Proof of Stake consensus protocol face considerable challenges when it comes to real–world application. PoW requires massive computation (a.k.a mining) to validate a new block. This process wastes an enormous amount of energy. PoS distributes new blocks based on a deterministic way, where the stake of a user is considered. Though more energy efficient, PoS–based blockchains face the potential of nothing–at–stake–attack, where bad behaviors are not punished. Notably, especially for some of the newly developed PoS–based blockchains, existing businesses or even individuals in the real–world can easily occupy the majority of the stake and render the entire blockchain compromised.

1.2 Ankr's Solutions

Ankr Blockchain is a revolutionary multi–chain and useful–work based blockchain with native authenticated data feed. Ankr Blockchain provides an innovative solution to the following problems:

● **Proof of Useful Work (PoUW)** allows for a self–sustainable blockchain framework with the potential to unlock idle computing power around the world, forming a decentralized **super–CPU**

● **Native authenticated data feed** with **standardized API** enables simple real–world business adoption

● **Multi–chain Plasma implementation** allows different applications to handle application–specific smart contracts on individual child chains, preventing transactions from overloading the PoUW–based main chain

We strive to build a resource–efficient blockchain framework that truly enables decentralized cloud computing and provides user–friendly infrastructure for business applications. Collaborating with industry partners, Ankr Network will also propose and contribute to several example child chains that address distinct business scenarios, including real estate, financial products, automobile, art collection, wine, and others.

Ankr will unlock new potentials with the following innovations:

● **Standardized APIs for connecting to existing business–ready Internet solutions**

● **Speed and scalability from a novel blockchain architecture**

● **Security and confidentiality enabled by trusted hardware**

● **Environmentally efficient mining and sharing of computing resources**

● **Reliable native oracle service**

2 Proof of Useful Work

2.1 Background

Bitcoin's underlying consensus protocol Proof of Work (PoW) addresses two fundamental problems in decentralized cryptocurrency design: how to select consensus leaders and how to allocate rewards fairly among participants. Even though Proof of Work (PoW) provides security to the blockchain, it nevertheless forces participants to waste a massive amount of computational resources. Other than preventing malicious attackers from taking over the majority of the mining resources, PoW as a consensus finds no other real-world benefit. As of the time of writing this white paper, the Bitcoin network uses more electricity than the entire country of Iceland and this amount is projected to reach as much as what is consumed by the population of Denmark by the year 2020.

Other popular consensuses such as Practical Byzantine Fault Tolerance (PBFT) or Proof of Stake (PoS) are essentially waste-free, but they restrict participation or require participants to lock in stakes of the blockchain. Further, PBFT and PoS consensus are often complex and arbitrary, making it extremely hard to update and adjust the original setup.

2.2 Trusted Hardware

Intel SGX (Software Guard Extensions) is a new set of instructions that permits execution of an application inside a hardware enclave, which protects the application's integrity and confidentiality against certain forms of hardware and software attacks, including hostile operating systems. An enclave processes isolated executions in SGX. These system calls are guaranteed to execute correctly and with perfect confidentiality.

SGX allows generation of authentication that remotely proves the validity of an operation. When an enclave is created, the CPU produces a hash of its initial state. The software in the enclave may at a later time request a report which includes a measurement and supplementary data provided by the process. The report is digitally signed using a hardware-protected key to produce proof that the software is running in an SGX-protected enclave. Such proof can be verified in a remote system and SGX signs proofs using a group signature.

2.3 PoUW Consensus Protocol

We propose Proof of Useful Work consensus, which is able to achieve a high-security standard without wasting energy. At a high-level, in PoUW consensus, miners with CPU computing power can execute useful computation under the trusted hardware's proctoring to earn reward within the blockchain ecosystem. In traditional PoW consensus, a reward can only be claimed after miners have solved complex hashing problems. Such a setup is due to the lack of a trusted proctor. SGX-protected hardware enclave can act as a trusted proctor for CPU activities, testifying miners' useful computation and providing proofs for mining reward.
The PoUW based Ankr Blockchain has a mining scheme composed of three components: miners, useful work providers, and the blockchain agent. Note that the blockchain agent is not a person, but an entity enabled by the trusted hardware.

The useful work providers will supply useful workloads to miners. The blockchain agent will collect transactions on the blockchain and generate corresponding block templates made ready once the proof of useful work is confirmed. In other words, in order for a block to be fully validated and published to the network, the block template requires a PoUW to be attached.

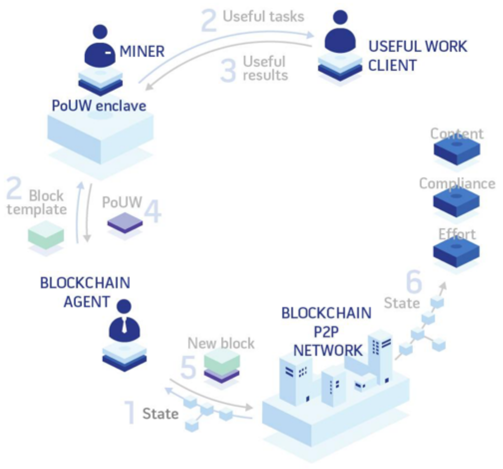
On the other hand, once the miner finishes its job, the hardware enclave verifies the validity of the miner's contribution and meters it based on a number of instructions. A complete PoUW consists of an attestation generated by SGX validating the compliance with Ankr and another attestation validating the matching difficulty parameter.

To decide which instruction deserves the reward, the PoUW enclave generates a random number and check whether this number is smaller than the desired difficulty. In this case, it uses SGX's random number generator (SRNG) to handle instructions in batches. Concretely, PoUW will divide useful workload into subtasks of much shorter duration. Each subtask will run to completion and compete for winning. If the result has a lower number of executed instructions than any other, the enclave will generate the attestation that includes the input block hash and difficulty.

Pseudo Code for the Miner Loop:

```
while True:
blockchain agent → block template;
process (template) → (hash, difficulty) ;
useful work client → task;
enclave (task, hash, difficulty) → PoUW;
if PoUW ≠ ⊥:
(block template, PoUW) → new block;
```

PoUW opens up new opportunities for the Ankr blockchain as miners' computing power within the network can be used towards almost all on-chain computation. For instance, if a client submits a private datagram request through our oracle service, the miner will use its resources to help encrypt and decrypt the parameters in the data-gram request. Excessive computational resources can be monetized and sold to internal or external applications for useful works. These useful work tasks can include training a neural network and even hosting web services.



In conclusion, to understand PoUW at a high level and compare it to existing solutions, we can imagine a world with only two things to do – computing and task-requesting. In a Proof of Work system, everyone computes essentially useless puzzles with no rewards, and for every predetermined time period, a lottery winner is randomly chosen (the most hard-working person has the best chance to win). In a Proof of Stake system, for every predetermined time period, a lottery winner is randomly chosen (the richest has the largest chance to win). And that one person is allowed to compute only on-chain transactions and get paid by the task requester. In a Proof of Useful Work system, everyone can compute all kinds of tasks that others request while getting paid by the requesters. With a trustworthy proctor monitoring everyone's computation actions, and for every predetermined time period, the proctor will randomly give out a reward to everyone who has done computing for others in this time period. People who perform more computing in this time period have a higher probability to get a reward. Of course, the "lottery winner" will be very happy with the extra reward, but everyone else will feel just fine as they also get paid. This is a system where everyone's endeavor gets rewarded, and the most hard-working individuals have the best chance to win the lottery.

With the widespread use of SGX–enabled CPUs, the PoUW protocol has the potential to unlock massive idled CPU (with SGX instructions) computing power for useful work computation. We envision that useful work can come from both computation tasks both on and off the blockchain. This essentially will enable distributed cloud computing, where otherwise idled CPUs can now generate value for owners with little cost. Further, under such distributed computing framework, the task requestors will have access to a cheaper cloud computing service compared to traditionally centralized cloud computing services. Currently, cloud service giants like AWS and Microsoft Azure charge extremely high markups.

2.4 Program Interpreter

We will be using Graphene SGX to provide developer interfaces for programming on the hardware enclave. Graphene SGX is a fully–featured library OS can rapidly deploy unmodified applications on SGX with overheads comparable to applications modified to use “shim” layers. Generally, Graphene–SGX interprets high–level code and converts them to C/C++. These supports can be runtimes or command–line, Apache, NGINX, GCC, R, Python, OpenJDK, etc.

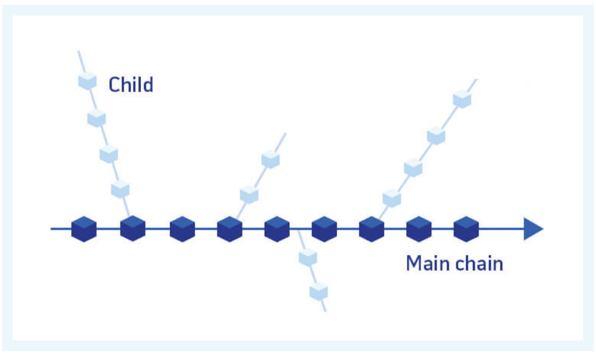
3 Multi-chain Structure

Ethereum processes all smart contracts on one chain in serial which bottlenecks throughput, especially when there are massive contracts and complicated data on the chain. We will look closely into how we plan to scale up Ankr blockchain through implementing Plasma and sharding.

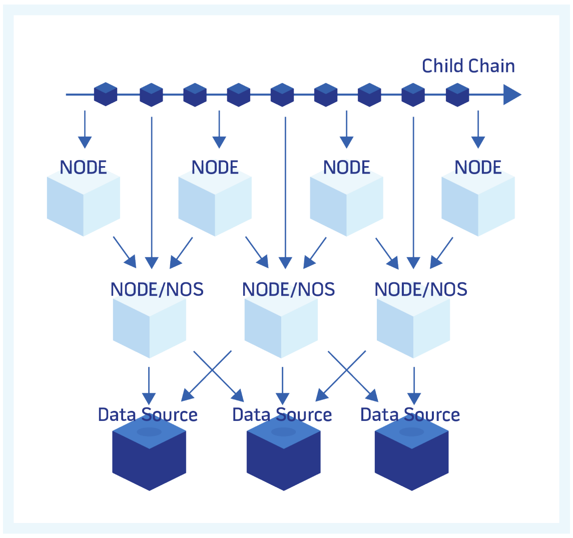
3.1 Chain Structure

Ankr network consists of a tree structure of blockchains where various application chains (Child Chains) are connected to a single root chain (Main Chain). Ankr's main chain is the backbone of the entire system. The main chain is based on PoUW and its functionality includes child chain indexing, useful work mining and smart contract executions. Each child chain is tailored to the need of a specific application. Basically, each block of the main chain contains the references to the boundary of child chains. As new blocks are generated on the child chain, the main chain will create new blocks to reflect the new boundary and cross-chain communication should be implemented as a message system to keep the isolation of chains.

Each child chain can be tailored to serve different purposes based on technical need. Application–specific smart contracts will be stored on the child chain, and the main chain will be used for consensus and distributed computation. Because the main chain can tap into a distributed global computing power, transactions on the child–chain will be calculated at a speed much faster than transactions on a traditional one–chain structure. Transactions can even be reversed if a participant within the child blockchain is proven to have acted maliciously.



The main chain will also provide a native authenticated data feed service for off–chain data to relay to each child chain. Currently, existing oracle solution exists separately from the blockchain framework and is limited in compatibility. We propose an user–friendly universal API for each child chain to connect to off–chain entities. Existing business can build decentralized autonomous applications on the child chain with powerful computing power and native data feed service provided by the main chain.



Different tailored needs can be categorized as follows:

- Low transaction volume but high transaction amount, eg. Real Estate
- High transaction volume but low transaction amount, eg. E-commerce
- Real-time requests and responses, eg. Prediction Market

3.2 Plasma

Plasma is a protocol for building unlimited number of blockchains that can are supervised by Ankr's main chain. The efficiency of the main chain can be significantly improved by offloading some transactions from the main chain to Plasma chains, especially if proper incentives are given to Plasma operators. Another advantage is the flexibility of Plasma chain implementation as long as it can be effectively cross-checked by contract on a parent chain. With new cryptographic approaches one can extend Plasma implementation with transactions utilizing ring signatures of zkSNARKs for confidentiality of end users.

Every participant in the network must run both a Plasma node and the Ankr node. Blocks must store all transactions and then form a 16-depth merkle tree with each transaction being a leaf (Unfilled leaves are zeroed out). The merkle root is then hashed together with the hash from the previous block header and stored as root hash. The block is then propagated to the network through gossip protocol and the root hash with timestamp is submitted to root chain through Ankr node with the validator address. There will be a way for validator to quickly check if UTXO is most recent. When someone makes a deposit, add the transaction to UTXO set. When someone makes valid transaction from a transaction output in the current UTXO set, we remove the original tx output from the UTXO set and add its outputs to the new UTXO set. When someone successfully exits, we simply remove the tx from a UTXO set.

This is how transaction must be stored in RLP serialized format.

```
[blknum1, txindex1, oindex1, sig1, # Input 1
blknum2, txindex2, oindex2, sig2, # Input 2
newowner1, denom1,                # Output 1
newowner2, denom2,                # Output 2
Fee]
```

Each transaction has 2 inputs and 2 outputs, and the sum of the denominations of the outputs plus the fee must equal the sum of the denominations of the inputs. The signatures must be signatures of all the other fields in the transaction, with the private key corresponding to the owner of that particular output. A deposit block has all input fields, and the fields for the second output, zeroed out. To make a transaction that spends only one

UTXO, a user can zero out all fields for the second input.

In the case where a participant sees an invalid block, they submit a fraud proof on the root chain. In the case where a participant does not receive a block, but they see a new block header hash submitted on the root chain; they must submit exit transaction as soon as possible. Also, they need to validate that the new block has same root hash as the root hash on root chain.

To send transaction, we will serialize transaction in format described above, propagate transaction to network and wait for the transaction to be included in a Plasma block. The block header hash will be added to smart contract on root chain

The more detailed implementation plan will include:

- **A list of Plasma blocks**, for each block storing (i) the Merkle root, (ii) the time the Merkle root was submitted. Create a Block struct which has both merkle root and timestamp recorded in each block. Mapping from uint256 block num to Block struct

- **A list of submitted exit transactions**, storing the submitter address and the UTXO position (Plasma block number, txindex, outindex). This must be stored in a data structure that allows transactions to be popped from the set in order of priority. Thi can be implemented using Priority Queue, with priority equal to blknum * 1000000000 + txindex * 10000 + oindex. Users should be able to challenge any exit within the priority queue (This requires being able to delete node from any place in priority queue).

In the future, we should be able to determine how to efficiently prove invalid blocks on chain. This allows users to prove a block is invalid, slash stake of owner and reward challenger. Block must then be rolled back by removing (blockNum, Block) pair from mapping of plasma blocks on root chain.

```
CONSTRUCTOR
● owner = msg.sender;
● owner_stake = msg.value;
● Blockcount = 1
● Create new PQ
      ○ PQ can only be called by contract
      ○ PQ must allow removal of node anywhere in data structure to allow for parallel challenges to
        take place.
```

- submitBlock(bytes32 root): submits a block, which is basically just the Merkle root of the transactions in the block
    - Add a mapping from (blockcount + 1, to struct(root, block.timestamp))
    - Can only be called by owner
    - Fire BlockSubmitted(rootHash) event

- deposit(): generates a block that contains only one transaction, generating a new UTXO into existence with denomination equal to the msg.value deposited
    - Root contract creates block and hashes it the way the child chain would and adds it to the plasma block list.
    - Validator must submit block with the next block number. If they try to submit block with same block number it will be ignored.
    - If depositor never sees the block with their deposit, they can still submit startExit with all parameters except for confirmSig
    - If they see the block with their deposit, but any future transaction is withheld, they again submit startExit with all parameters except for confirmSig
    - In the case where they send the transaction, its included in the block but that block withheld. They can still successfully exit because any challenge needs a confirmSig from sender saying have the block.
    - Fire Deposit(msg.sender, msg.value) event

- startExit(uint256 plasmaBlockNum, uint256 txindex, uint256 oindex, bytes tx, bytes proof, bytes confirmSig): starts an exit procedure for a given UTXO. Requires as input (i) the Plasma block number and tx index in which the UTXO was created, (ii) the output index, (iii) the transaction containing that UTXO, (iv) a Merkle proof of the transaction, and (v) a confirm signature from each of the previous owners of the now-spent outputs that were used to create the UTXO.
    - When we just have a fresh UTXO that needs to exit, we put 0x0 in for confirmSig parameter
    - When we have an old UTXO, we need aggregate of all signatures from previous owners (How do we aggregate?). Figure out if OmiseGo does aggregate signature.
    - Merkle proof on the transaction, validate that it corresponds to hash of plasmablocknum on list of plasma blocks
    - This needs to be bonded.
    - Only the user who received this transaction should be able to call this method (no one can exit someone else's transaction)
    - Fire ExitStarted(msg.sender, plasmaBlockNum, txindex, oindex) event.

- challengeExit(uint256 exitId, uint256 plasmaBlockNum, uint256 txindex, uint256 oindex, bytes tx, bytes proof, bytes confirmSig): challenges an exit attempt in process, by providing a proof that the TXO was spent, the spend was included in a block, and the owner made a confirm signature.
    - ecrecover(hash(tx), v, r, s): address returned must be the same as sender of exit transaction.
    - Merkle proof on the transaction, validate that it corresponds to hash of plasmablocknum on

list of plasma blocks
    - This needs to be bonded.
    - Bond submitted by exiter in startExit is reward to challenger if successful. Allows one to delegate challenging of improper exits to third party.

- finalizeExit()
    - Pops exit off of PQ if challenge period ended
    - Pays exiter required money
    - Note: can be called by anyone.
    - Fire ExitFinalized(address exiter, plasmaBlockNum, txindex, oindex) event

- withdraw()
    - Transfers any owed funds to msg.sender (includes exits and refunded bonds)
    - Must have way of keeping track of the funds bonded by msg.sender and whether that bond can be released.
        - Create mapping(address => uint256) withdrawals. Upon successful exit, increment with drawals[msg.sender] with exit funds + bond. Upon successful challenge, increment withdrawals[msg.sender] with bond of exiter + own bond.

Blocks can only be sent one by one, in a sequence enforced by the contract. Any user of Ankr network can deposit Ankr tokens to a contract that will trigger an event and force Plasma network operator to make a fund–ing transaction in Plasma chain. Then the users can make transactions freely in the Plasma chain with only headers being pushed to the parent contract on Ankr.

When a user would like to settle one of his transactions in the main chain, the user can make a withdrawal on Ankr by providing a reference number to the transaction. It will require the full transaction and Merkle proof to prove if this transaction is valid in the blockchain context and the user is a legitimate owner. Parent contract checks a proof and if it passes, the withdrawal starts.

4 Native Data Feed for Smart Contract

Smart contracts are programs that execute autonomously on the blockchain without external connectivity. Their key uses (e.g. financial instruments) require them to consume data from outside the blockchain (e.g. stock quotes). Today, almost all blockchains are data-hungry. In order to build a sufficient ecosystem for smart contracts to be widely adopted, a trustworthy data feed system that supports a large amount of data requests is extremely critical.Therefore, a mechanism such as an oracle service is required to bridge the two environments: on-chain and off-chain.

Again, considering speed and scalability as our design goal, this oracle service must be reliable and efficient. We introduce Ankr's native data feed, which is enabled by the Native Oracle System – an authenticated data feed system using trusted hardware.

4.1 Undesirable Existing Oracle Solutions

● Centralized Oracle: A centralized oracle is the most commonly used approach in many blockchains. Such a system is quite the antithesis of decentralized blockchain and smart contracts as it fails to provide tamper resistance and trustlessness. Almost all centralized oracles rely on off-chain notarizations, which can potentially create troublesome results.

● Manual Human Input: Many proposals focus on full adoption of manual human input. Though decentralized and flexible, such an approach is not only time-inefficient, but also resource-intensive.

● Maneuvering data sources: Even though TLS-N provides digitally signed data sources, such a method would require all legacy systems and websites to change their infrastructure accordingly.

4.2 NOS Components

A Native Oracle System consists of three components: **the NOS smart contract, the Enclave and the Relay**. The Enclave and Relay reside on the NOS server and the NOS smart contract is executed on the blockchain.

The NOS Smart Contract functions as the frontend of the NOS system. It provides an API to interact with ANY form of smart contract such as the Ethereum Smart Contract written in Solidity. Concretely, a NOS Smart Contract accepts a datagram request from a blockchain-based smart contract and responds with the corresponding datagram. CNOS also provides monetary management similar to Ethereum's gas. Additionally, it fixes many existing issues with the Ethereum's gas, which will be addressed later.

The Enclave is responsible for ingesting datagram requests from blockchains. It queries external HTTPS-enabled internet data sources, returns a datagram as a digitally signed message. The Enclave should be assumed secure as it lacks network access and is completely isolated from any operating system and software, whether hostile or not.

The Relay handles bidirectional network traffic on behalf of the Enclave.

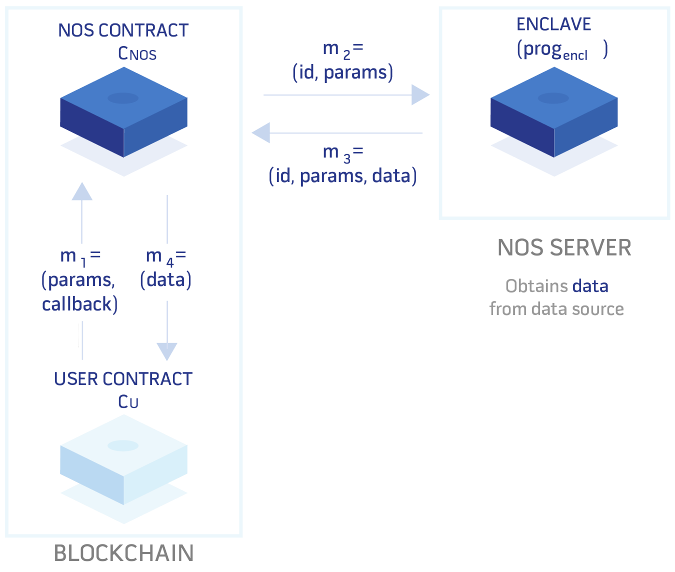
4.3 Data Feed Protocol

● Initiate data request: the smart contract sends a datagram request to NOS Contract

● Monitor and relay: the Relay monitors the NOS Contract and delivers any incoming request to the Enclave.

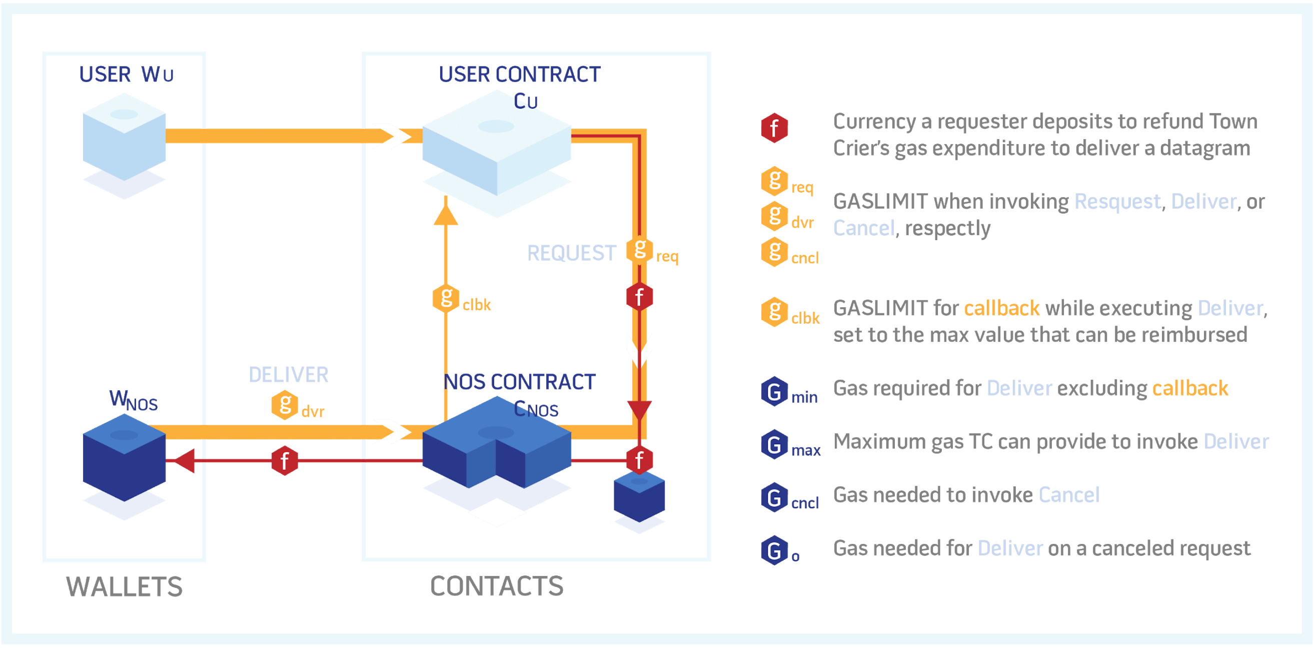
● Fetch data feed: the Enclave communicates with the data source via HTTPS with the given parameters. After obtaining the datagram, it forwards the datagram to the NOS Contract via the Relay.

● Response: NOS Contract returns the datagram to the smart contract.



4.4 Improvements in Security and Confidentiality

● Gas sustainability: Ethereum requires transaction initiators to pay gas costs. Such a design has the risk of malicious users triggering calls to steal gas, causing the depletion of gas and potential denial–of–service attack on the application level. It is therefore critical that any gas services on blockchain can be reimbursed for computation initiated. One significant drawback of the current Ethereum gas model is that, if a transaction is submitted by a wallet with insufficient funds, the wallet's balance will drop to 0. NOS Protocol ensures that an honest system will not run out of money and that an honest requester will not pay excessive fees.


USER W U
USER CONTRACT C U
REQUEST g req
f
g clbk
DELIVER g dvr
W NOS
NOS CONTRACT C NOS
f
f
WALLETS
CONTACTS
f Currency a requester deposits to refund Town Crier's gas expenditure to deliver a datagram
g req g dvr g cncl GASLIMIT when invoking Resquest, Deliver, or Cancel, respectly
g clbk GASLIMIT for callback while executing Deliver, set to the max value that can be reimbursed
G min Gas required for Deliver excluding callback
G max Maximum gas TC can provide to invoke Deliver
G cncl Gas needed to invoke Cancel
G o Gas needed for Deliver on a canceled request

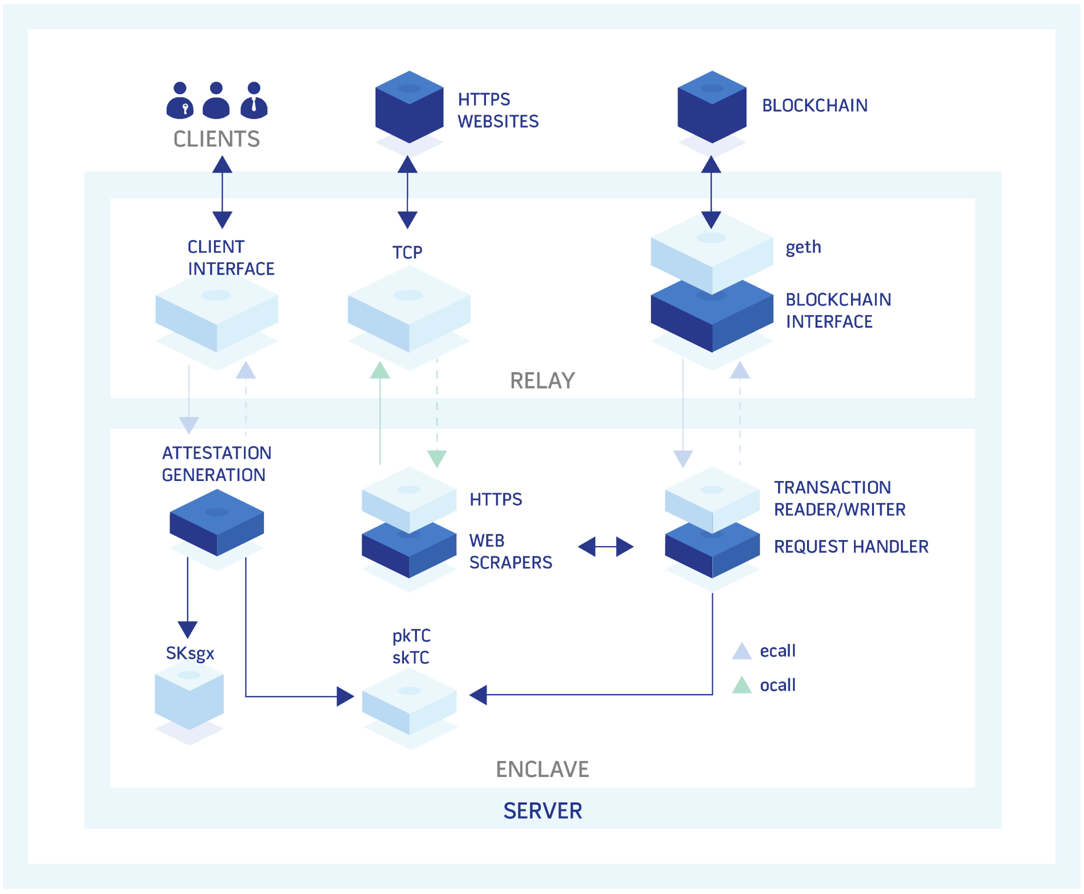
● Private datagram requests: Not all transactions have to be publicly visible. NOS supports permission–based user confidentiality by encrypting parameters in the datagram requests. Only users with permission will be able to see the data responses.

4.5 Use Cases and Unlocked Potentials

● HTTPS–enabled Website and Legacy Systems: NOS can easily transfer data from HTTPS–enabled websites to the blockchain. For example, MLS is a trusted source for all real estate data. The API supports data of recent sales, public and private schools, demographics, home values and market trends. NOS can deliver the data in bulk in JSON or XML onto the blockchain.

● Potential fiat currency transactions on blockchain: NOS should be able to provide a bank–intermediated ledger through commercial banks' API. By design, it should present transferring proofs, making purchases of smart assets with fiat currency possible.

● Potential modern application interfaces: NOS system can connect blockchain with modern applications such as Facebook messenger and Wechat to unlock massive mar–keting potentials.



5 Ecosystem on the Ankr Blockchain

5.1 Ankr Token System

Ankr tokens serve as the storage of value and as a means to transfer value. At all times, Ankr token can be used to incentivize the engagement with Ankr network and can be transferred across child chains. Since it is more complicated to finish transactions within a multi-chain system, the main chain, with global knowledge, will be the best place to maintain the allocation of tokens generated.

Ankr tokens can be used as computation fees across all child chains, and all users of Ankr blockchain can earn Ankr tokens by contributing their computing power. Such ecosystem is a virtuous cycle, as the more people participate in Ankr blockchain computation, the more Ankr token they will earn, the more token they have, the more service they can acquire in the ecosystem, and thus triggering more computation needs.

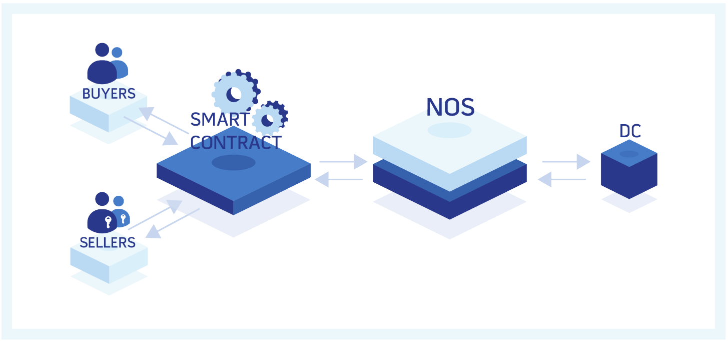
Ankr tokens can be mined by contributing idle computing power for useful work computation and will be charged for computation and transactions happened on Ankr Blockchain. This will create a truly self-sustain-able ecosystem and create a "decentralized world computer." The decentralized computer will cost users Ankr token to use, but it will be much cheaper than centralized solutions such as Amazon cloud or Google cloud, because the decentralized cloud will utilize otherwise wasted computing power and wouldn't charge high markup like the internet giants.

Moreover, human capital for running the decentralized computer will be much lower, as the decentralized solu-tion won't spend a large amount of money for administration, marketing and high executive compensation. In other words, the "decentralized computer" will provide the strongest computing power with the lowest cost.

We envision the Ankr blockchain to be a truly self-sustainable ecosystem with growing popularity among the community and ever-increasing computational power.

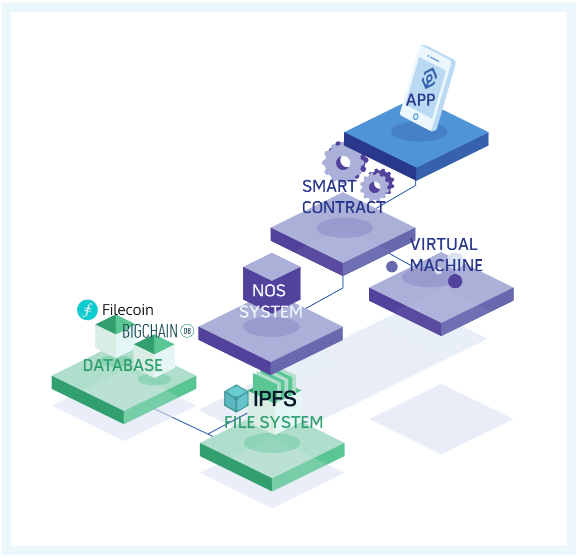
5.2 Smart Identity and Credit System

With the embedded oracle service on the Ankr Blockchain, users will be able to build their own identities as smart contracts within the ecosystem. In the future, instead of being restricted to one dApp, users can interact with all dApps on the Ankr Blockchain. One user exists singularly on the chain with the smart contract recording all the user events on the blockchain. Off-chain assets can also be digitalized and tokenized and be written into smart contracts, with an immutable log of all transactions. The connected internet of things enables a truly innovative credit system that allows both sides of the transaction to be transparent and untampered.

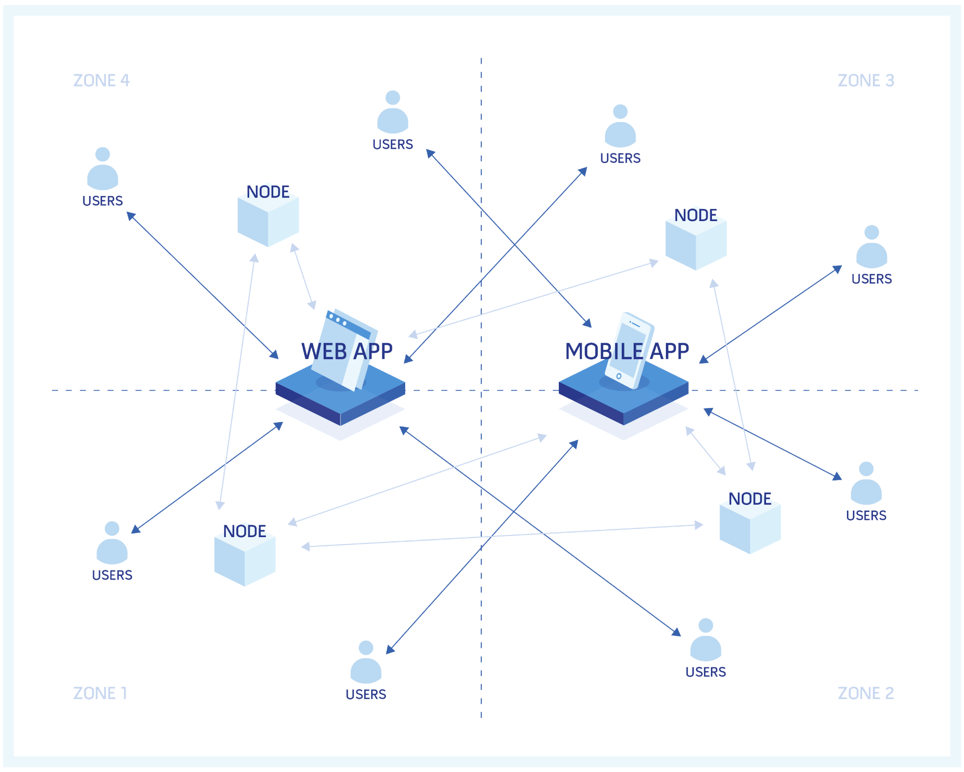


5.3 DApps and Programmable Interfaces

A well-designed public API is crucial for business adoption and usability. Let's take Amazon AWS as an example. The Simple Storage Service (S3) provides operations like retrieve, store, refresh, update and etc. Most customers do not care or need to understand how their data is duplicated and distributed for the purpose of resilience and accessibility. Similarly, a blockchain app developer is more interested in promoting assets on the chain, removing assets off the chain and maintaining the status. We will provide more information about our long-term plan in the section on ecosystem and dApp.



5.4 Context within the Distributed Cloud



6 Use Cases

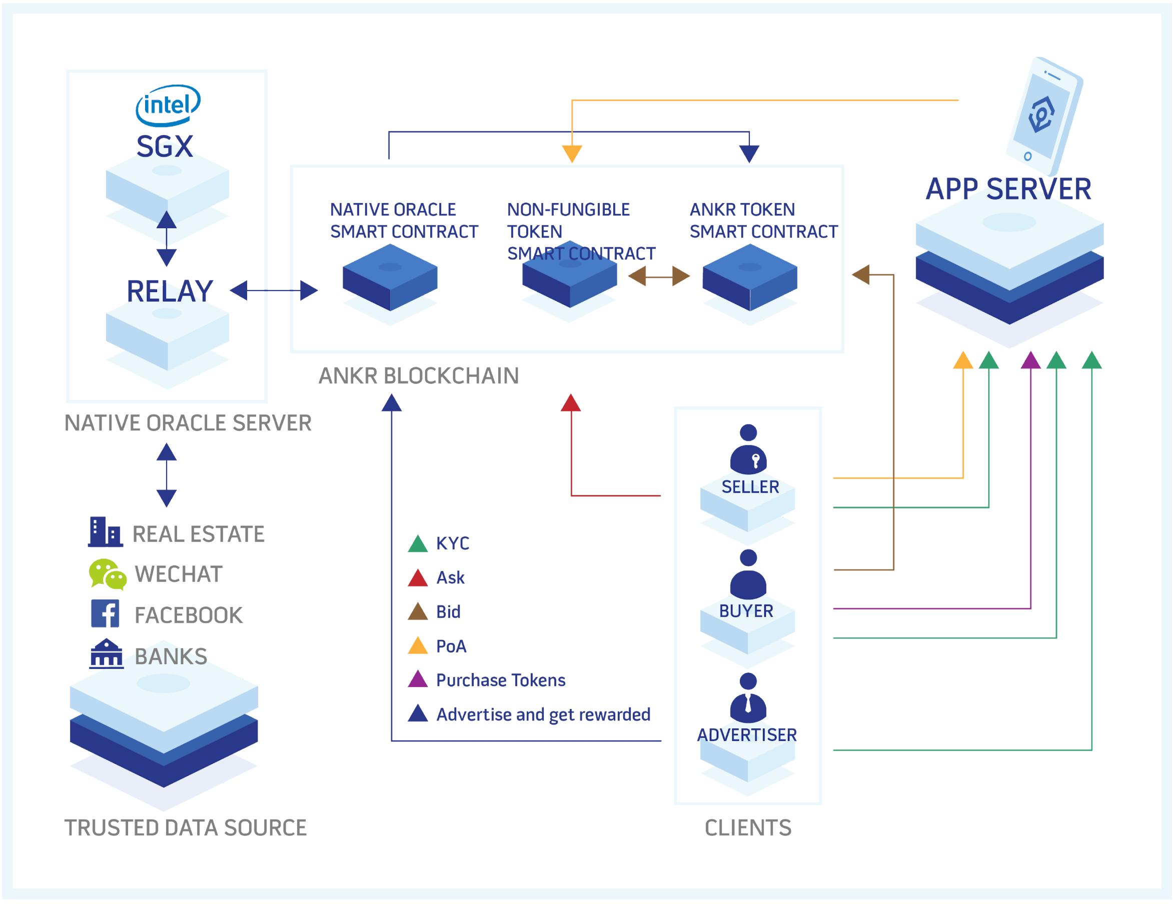
Ankr is working closely with various partners across the real estate, fintech, automobile and art collection industries for future Dapps on Ankr blockchain. All Dapps on the Ankr Blockchain will use Ankr tokens for computation and transaction fees. We believe with native oracle services and a resource efficient consensus, the Ankr blockchain will be an ideal platform for existing businesses to build their blockchain applications.

6.1 Real Estate

The real estate industry is characterized by asymmetric information and heavy offline operations. All parties in the industry try to create an edge for themselves via the use of asymmetric information, by concealing information that could positively hurt their position in the transaction. Due to such dispersed and centralized information ownership, countless manual information exchanges are required, and in turn, opportunities to tamper with information are created. As a result, massive friction and ambiguity persist in the industry.

The Ankr blockchain provides a turnkey solution to this intransparent and inefficient industry, as the Native Oracle System can automatically update off chain data, and private datagram requests can support permis–sion–based information visibility. We believe full information visibility to all users on the network isn't ideal for a real estate application, as sensitive information such as one's rent payment records or property's income history should only be made visible to counterparties when permission is given. With this logic in mind, Ankr Networks and our partner, CitySpade, strive to build the first native Dapp that provides a blockchain based rental payment and rental contract marketplace that will form a personal rental credit system over time. Moreover, with an established rental credit system, crowdfunded security deposits, tokenized rental agree–ment transactions or even partial real estate ownership trading can become a reality.

More than 6,000 apartments managed by CitySpade across New York, Philadelphia, Boston, Washington DC and New Jersey will become the first wave of properties that are given "smart identities" that record all future transactions and rental income of the underlying properties. More than $300 million worth of annual rental payments will be processed in the native application right from the beginning, and we believe more properties as well as users will join the platform for ample incentives created organically by the platform and an unprece– dented automation experience. (Detailed description see Real Estate Native Dapp Whitepaper)



6.2 Asset–backed Securities

The Ankr Blockchain, with its high computation capability and native real–time oracle feed, can largely streamline the asset–backed security origination processes, lower costs, increase the speed of transactions, enhance transparency, and fortify security. Securitization services based on the Ankr blockchain could impact all participants in the securitization lifecy–cle—from originators, sponsors/issuers, and servicers to rating agencies, trustees, investors, and even regulators.

Along with our partner, Ankr Network is proposing a child–chain application that rein–vents the securitization lifecycle (here we use a loan–backed security as an example).

- Starting with the loan origination. When borrowers and lenders agree to the terms of a loan, a new electronic asset will be created on the blockchain and time–stamped. Ownership information and other pertinent underwriting data will be attached to the loan. Smart contracts with relevant loan information will enable automatic loan servicing and records of the borrower's payment history.
- A group of loans will have the ability to be pooled together into an SPV, where individual servicing history will be linked to the original borrower. This allows future secu–rity holders to view the underlying loan's performance in almost real time. When a loan–backed security is created, relevant legal and offering documents will be automatically created by smart contracts with little to no manual assis–tance. Regulators will be given full permission to all rele–vant information for compliance, while the public will only have limited information access for this digital security. Most importantly, smart contracts will automate the distri–bution of payments to the security holder, giving investors real–time visibility to their income stream without waiting and without manual actions.
- Rating agencies, regulators and traders will have differ–ent permission to the same data that ties to the underlying assets. Such data will also be updated in real time for all users, giving participants in the market a single trust–worthy asset information source.

References

[1] I. Eyal, A. E. Gencer, E. G. Sirer and R. V. Renesse. Bitcoin-NG: A Scalable Blockchain Protocol. NSDI, 2016.

[2] ZHANG, F., EYAL, I., ESCRIVA, R., JUELS, A., AND VAN RENESSE, R. REM: Resource-Efficient Mining for Blockchains. Cryptology ePrint Archive, Report 2017/179, 2017.

[3] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi. Town crier: An authenticated data feed for smart contracts. Cryptology ePrint Archive, Report 2016/168, 2016.

[4] Intel Corporation. Intel® Software Guard Extensions Programming Reference, 329298-002us edition, 2014.

[5] Intel Corporation. Intel® Software Guard Extensions SDK. https://software.intel.com/en-us/sgx-sdk, 2015.

[6] Intel Software Guard Extensions Enclave Writer's Guide. https://software.intel. com/sites/default/files/managed/ae/48/ Software-Guard-Extensions-Enclave-Writers-Guide. pdf. Accessed: 2017-2-16.

[7] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.

[8] V. Buterin. Ethereum: A next-generation smart contract and decentralized application platform.

[9] Gavin Wood, Ethereum: A secure decentralised generalised transaction ledger, http:// gavwood.com/pa per.pdf, 2015.

[10] Oraclize: "The provably honest oracle service". www.oraclize.it, Referenced Feb. 2016.

[11] E. Buchman. Tendermint: Byzantine Fault Tolerance in the Age of Blockchains, 2016.

[12] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. In 3rd USENIX Symposium on Operating Sys tems Design and Implementation (OSDI), Feb. 1999.

[13] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, Bryan Ford. 2017. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding

[14] Randhound and Randherd. https://github.com/dedis/paper_17_randomness.

[15] V. Buterin. J. Poon. Plasma: Scalable Autonomous Smart Contracts