Cellframe:
Framework for blockchain integration secured by post-quantum cryptography

Version 1.0 (draft)
October 2019

Authors
Alexey Vesnin, Vasily Sumanov, Yardy

Abstract
In this document, the draft version of Cellframe architecture is presented. The Cellframe is a framework for blockchain bridging secured by post-quantum cryptography and aimed at providing seamless interchain operation service with a high level of security. Cellframe can be used as blockchain infrastructure for launching distributed web services (VPN, CDN, video streaming) as well. The core components, such as ZeroChain, Shard, AURA (blockchain bridging module) and inter-shard communication are described. The post-quantum cryptography implementation is not a topic for this paper: full materials can be found in wiki[1].

Contents

1. Introduction
Cellframe is an open-source framework for blockchain bridging secured by post-quantum cryptography. Based on original sharding implementation, Cellframe can provide extremely high transaction throughput and allow interchain communication. Besides interchain services, Cellframe supports the creation and operation of distributed networks - special integration between web services and blockchains. Distributed networks are presented by distributed versions of traditionally centralized internet services such as VPN, CDN, cloud (fog) computing, and video streaming.

2. Cellframe

[1] wiki.cellframe.net

The Cellframe architecture is based on strict standardization, but only at those places and things that are required to be locked up that hard. The consensus algorithm has to be a pure computational and based on the strict mathematics that can be performed one way everywhere, so we're using a PoW based consensus on our main chain called ZeroChain and in the shard's fuel chain to ensure the equal rights and abilities to earn a reward for all the nodes interested, and - to make hashrate-based attacks harder to perform - we are using ASIC-resistant hashing function, so everybody can use their CPU and GPU resources to participate in mining process. However, we do have a node role, so it's a time-proven approach to make some nodes a masternodes. It also enables us to make a safety deposit be locked to protect from fraud attacks on a network. And also - adding a healthy feature from the PoS approach - we're adding a staking ability. Staking can be used as a DarkSend built-in mixer basis and also as a scarcity factor for tokens involved. An additional use for a staking process is a safety deposit that can be slashed in case of providing false answers or misbehaving, but the details will be disclosed further as the precise functional parts will be described. Both PoW and PoS are based on strict mathematical rules that can be only calculated, compared and evaluated as a number, we have no logical overlays to avoid both overcomplexity and edge task cases vulnerabilities for a logical scheme itself. Speaking of the phases for PoW/PoS consensus - we do not have ones, this mechanism looks like a counter-productive in our particular case: the consensus will always be based on a fixed rules, PoW will always be the sole consensus basis and PoS will help us to finalize blocks faster and to determine and protect the nodes' roles.

3. Architecture and implementation

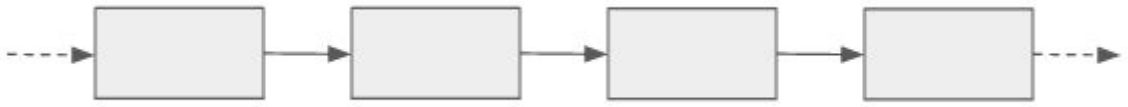
The principal architecture of Cellframe consists of several logical parts:

- ZeroChain
- Shards
- AURA (Bridging service, runs on top of a shard)

The logical scheme for ZeroChain and shard's fuel chains is based strictly on a single code base and the difference is in a constructor's argument of a boolean type - if it's true, then we're running a ZeroChain, if it's false - we're running a fuel chain. This strictness in implementation provides us an ability to fix bugs as wide as possible, so it won't affect any functionality at all. Both scenarios are involving tokens, but no smart contracts are in there due to the speed and throughput requirement: yes, we do have a lot of functionality of the above-coin level, but it all implemented as native code and finalized at node compilation time.

ZeroChain

ZeroChain is a blockchain with a typical block structure, similar to DASH and PIVX. It is secured by a hybrid PoW/PoS consensus with a native CELL token.



ZeroChain can be considered as main registry of the system and stores the following information:

- GUUID (global ID for any object in the system - chain, token, smart-contract etc)
- Public keys of masternodes and masternode announcement events
- Governance (propose, vote, rewoke)
- Binary blob for storing any information from shards
- CELL tokens balances
- Global DEX for fuel tokens

Proof-of-Work is implemented as an ASIC-resistant algorithm based on variable Keccak hash function which requires a significant amount of memory and can be calculated only on GPU or CPU.

Proof-of-Stake is implemented similar to DASH/PIVX but contains two levels of masternodes. The level 1 (M1) masternodes are involved in transaction signing and insta-TX (instant send) service, while level 2 (M2) masternodes have an additional option to launch a new shard in the ecosystem. A staking amount for M2 level is an order of magnitude above the M1 - like ten times or more. The reason for this difference is not just a way more significant role of the M2 masternode, but the demand for making a M1 masternode affordable for a wide range of investors.

Masternode M1 can be launched from one address by sending locking transaction with necessary number of CELL tokens. Masternode M2 can be launched like M1 masternode or as multisig wallet by accepting contributions from several addresses. Thus, M2 node launch and following shard initiation can be carried out without a single point of key storage and responsibility for shard operation - if it’s a requirement. However, a remark for security alert - it is not imposed as a full solution for key protection problem, because every participant of a multisig wallet has to protect it’s own private key with exactly the same strict security measures and protocols, it’s only a decent single point of failure elimination mechanism, no more!

Each object in the Cellframe ecosystem has to have GUUID, announced in ZeroChain to be available for usage across Cellframe shards and connected blockchains.

Masternode announcement workflow

The M1 masternode is announced by a standard routine:

1. A single transaction with an amount of stake lock is made to the PK/wallet that is supposed to become a masternode.
2. A transaction hash from step 1 is used in the node configuration process as a masternode initialization key in conjunction with a line masternode=1
3. As the transaction has 100+ block depth, i.e. it's more than a 100 blocks mature, so all the active network nodes will have the block with it for sure, a masternode is started and is announcing itself as a masternode and at that moment a service message with the TX hash is broadcasted across the network.

So in time, the masternode will be selected for participating in a block finalization process and will receive the reward for that. If not elected but active, it still participating in the blockchain consensus.

The M2 masternode is started in a way more complicated routine due to its role. The key difference is that if for an M1 masternode all we have to do is to lock a stake and we need to do it once, for M2 masternode we do have a need to amend it's detailed as new M2 nodes will rise because cross-staking forces the existing shards to stake up the coins of the new ones lately, so the new wallet addresses have to be published in a zerochain. So it goes as follows:

1. The announce service message is imprinted inside a zerochain: it contains the Public key(s) for signing the shard-related data, weight, and quorum for a multisig if it's used, a list of wallets for all the active M2 shards' fuel tokens at the moment of creation/announcing.
2. If the amount of funds in fuel tokens is not enough, a monitoring phase begins: all other M2 shards are periodically checking that the balance increase is above the pre-defined ratio for all the fuel tokens on the new M2 node's accounts, and if this rule is broken - the announce is treated as invalidated and marked as one with a signature of all the active M2's.
3. If the staking requirement is fulfilled - then all the active M2's are signing the activation message in zerochain and the M2 node and its shard is becoming active from this very moment and its fuel token blockchain is started. Also, as this message is propagated - all the members of the M2 node, if any, are opening their p2p endpoint listener for shard-related service requests. Also on DEX on zerochain the new fuel token is automatically listed and an initial amount of it is put on a trade.

4. The active M2 nodes are automatically opening their wallets in the fuel token of a new shard and are amending their descriptors in zerochain blockchain with a wallet addresses.
5. Monitoring is started for an active M2's to stake up the fuel token of a new shard - with exactly the same criteria as a new node was monitored in step 2. The channel of obtaining the tokens is fully up to the shards' owners - a global DEX is just for convenience purposes only, it's not a mandatory channel.

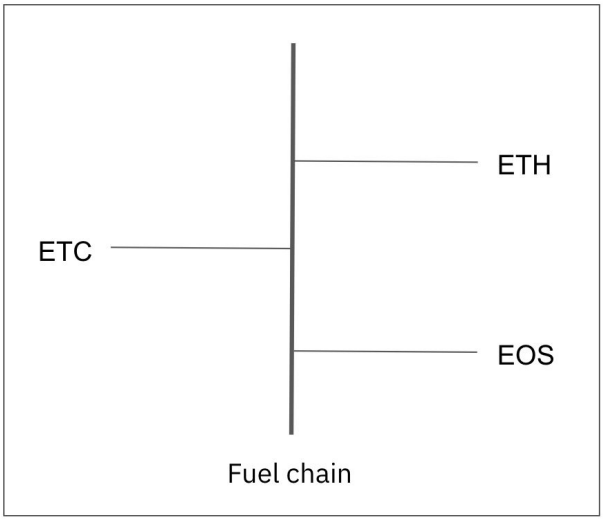
Network Nodes

There are several basic types of nodes in the Cellframe. All types are identical for ZeroChain and shards.

By type	By role	By memory requirements
Regular Node	Normal node: bdb from the last checkpoint	Normal (standard caching)
Master Node	Full node: bdb from the genesis block	Huge (max caching)
		Light (min. threads+Mempool, BerkeleyDB)

Shards

Basic architecture of Cellframe shard is equal to ZeroChain and uses the same library toolkit for maintaining and running blockchain. Each shard is secured by its own fuel chain with PoW/PoS consensus. Each shard is an isolated logical unit, connecting several blockchains (public or private) to interoperability sub-network by means of AURA (a tool for inter-blockchain bridging, integrated into Cellframe framework).



Cellframe shard

The M1 masternode is announced by a standard routine:

1. A single transaction with an amount of stake lock is made to the PK/wallet that is supposed to become a masternode.
2. A transaction hash from step 1 is used in the node configuration process as a masternode initialization key in conjunction with a line masternode=1
3. As the transaction has 100+ block depth, i.e. it's more than a 100 blocks mature, so all the active network nodes will have the block with it for sure, a masternode is started and is announcing itself as a masternode and at that moment a service message with the TX hash is broadcasted across the network.

So in time, the masternode will be selected for participating in a block finalization process and will receive the reward for that. If not elected but active, it still participating in the blockchain consensus.

The M2 masternode is started in a way more complicated routine due to its role. The key difference is that if for an M1 masternode all we have to do is to lock a stake and we need to do it once, for M2 masternode we do have a need to amend it's detailed as new M2 nodes will rise because cross-staking forces the existing shards to stake up the coins of the new ones lately, so the new wallet addresses have to be published in a zerochain. So it goes as follows:

1. The announce service message is imprinted inside a zerochain: it contains the Public key(s) for signing the shard-related data, weight, and quorum for a multisig if it's used, a list of wallets for all the active M2 shards' fuel tokens at the moment of creation/announcing.
2. If the amount of funds in fuel tokens is not enough, a monitoring phase begins: all other M2 shards are periodically checking that the balance increase is above the pre-defined ratio for all the fuel tokens on the new M2 node's accounts, and if this rule is broken - the announce is treated as invalidated and marked as one with a signature of all the active M2's.
3. If the staking requirement is fulfilled - then all the active M2's are signing the activation message in zerochain and the M2 node and its shard are becoming active from this very moment and its fuel token blockchain is started. Also, as this message is propagated - all the members of the M2 node, if any, are opening their p2p endpoint listener for shard-related service requests. Also on DEX on zerochain the new fuel token is automatically listed and an initial amount of it is put on a trade.
4. The active M2 nodes are automatically opening their wallets in the fuel token of a new shard and are amending their descriptors in zerochain blockchain with a wallet addresses.
5. Monitoring is started for an active M2's to stake up the fuel token of a new shard - with exactly the same criteria as a new node was monitored in step 2.

The channel of obtaining the tokens is fully up to the shards' owners - a global DEX is just for convenience purposes only, it's not a mandatory channel.

P2P workflow

As mentioned before, p2p inter-shard messaging is one of the key features of cellframe, so here is how it works:

1. Shard A makes a prepayment in Shard B's fuel and receives a mature TX hash
2. Shard A queries the zerochain to get Shard B endpoint addresses
3. Shard A requests JSON-RPC to shard B with a TX hash from step 1 and receives a ticket ID for an answer
4. Shard B accepts the request and closes the TCP connection
5. Shard B makes the prepayment in Shard A's fuel and receives a mature TX hash with exact matching maturity conditions as in step 1
6. The result is delivered from shard B making a connection just like shard A did to deliver it's request.

So, the fuel tokens are circulating, fuel chains are the same worth to be mined as the zerochain. And it also incentivizes the shard owners to make a working integrations that will be demanded high and used regularly.

AURA

The AURA bridging tool is one of the key components of the Cellframe ecosystem. It was developed with a goal to create a blockchain communication network on the basis of each shard.

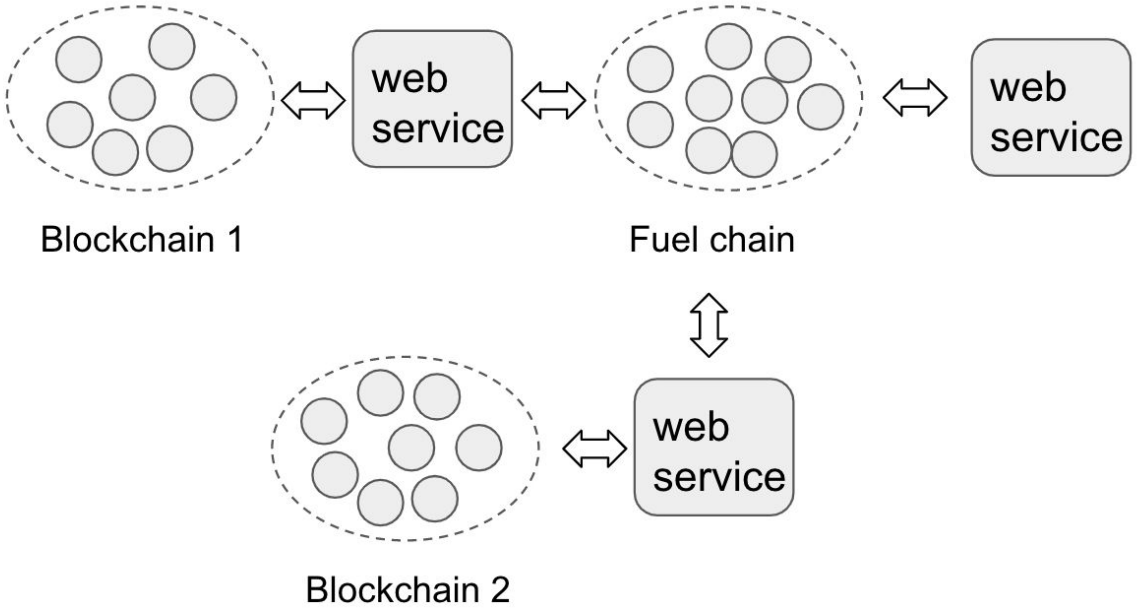
The AURA solves a problem of secure blockchain bridging. Centralized bridges are compromised initially as one person controls frozen funds and can steal them or interfere in bridge operation. The latest big case of hacking a centralized bridge is Fusion which resulted in losses of $6.5 mln.

Basically, AURA allows two blockchains to carry out simple operations like creation token derivatives by freezing original assets on one blockchain and printing it derivative (in the meaning of "mirror copy") on another one.

The principal AURA service workflow:

Initially, there are two blockchains and webservice for communication.

From each blockchain side, a set of K validators are determined. For being a validator each node has to stake N native fuel tokens of a shard. Node is not obliged to be a shard masternode.



The external chain validation webservice monitors JSON-RPC of both validated chain and fuel chain. To become a validator, the node needs to have a synchronized copy of the validated chain, being announced and activated (activation and operation include tokens staking).

Each request consists of:

1. GUUID of the object to be checked out in read-only mode;
2. A minimal number of validators, that have to be active for this request. In case of no necessary amount of validators web service would return NO_QUORUM;
3. An opcode and its arguments;

The examples of possible requests:

- Get a block and its full data
- Get transaction with its full data and block reference
- Get the address and its balance
- Subscribing for an event in target external blockchain for a fixed period
- Monitoring a target external blockchain for a predefined condition for a fixed period with notification through fuel chain

In case of a request from blockchain1 to blockchain2, at least P validators subset from initially registered K validators is selected to perform the read-only operation and provide an answer. All individual answers are collected and the final result is sent to the bridging webservice.

The validators have economic incentives to behave honestly - each honest answer of a validator is rewarded by billing a requester, while each malicious answer results in

partially deposit slashing (a penalty). The criteria of “honesty” and “dishonesty” is the resulting answer of the majority of the nodes. The penalty fee is an order of magnitude above the correct answer reward and is not less than 0.1 of the deposit.

The query is not supporting pipelining to prevent spam with bulk requests: one query returns result for only one object. All queries are asynchronous.

Inter-shard communications

To allow seamless and load-balancing inter-shard communication, each shard has an endpoint that is designed for outside serial asynchronous JSON-RPC p2p queries and they are described above.

Another way of inter-shard communication is using a zerochain block with the transaction from one shard’s wallet to another one’s one with all the data encapsulated in a transaction comment. However, the comment field is small, so data can be put into the blob part before the request and be referenced in the message.

Each query is paid in native (fuel) token of a shard, performing the query and is similar to ordinary shard transactions: the key difference that the query is accepted via service requests from outside with prepayment’s TXO in a fuel chain. The query is executed in p2p mode, or record some data on the ZeroChain. The query structure is the same as the previously mentioned AURA query. The answer is returned in the same way as the request came in vice versa mode: it is paid in the fuel of the receiving party.

Also, when the request is accepted by a shard from outside, it provides an additional amount of tokens for backward flow. This mechanism ensures and incentivizes creating really, real-world applicable integration that will actually work on the market.

For network security and more effective inter-shard communication, the cross-staking rule was implemented.

Cross-staking rule

To launch a new shard it is mandatory to stake not only CELL tokens (that are required to launch M2 node), but also to stake tokens of **all** public shards that are active at the moment of the new shard announcement. In opposite, each public shard has to stake tokens of the newly launched shard (after the staking confirmation event of a new shard).

This rule was introduced due to the following reasons:

1. To increase overall network security with each launched shard;
2. To provide storage for paying inter-shard communication requests during the interaction between shards;
3. To make cross-shard integration deeper and interoperable;

Shard lifecycle mechanics

The mechanism of the process is the follows:

1. One or several ZeroChain addresses launch M2 masternode, the process is described above
2. The shard starts operation as automatically, as possible - it's all built-in inside the node code
3. In time, shard can make it's cross-staking wallets a masternodes in another fuel chains and earn a reward in the foreign fuel tokens. It's a handy possibility to earn some fuel for a future before it will be required to make an external request.
4. Shard's own wallet(s) for a request - it has to be a masternode in it's fuel chain, and a half of it's reward is automatically sent on DEX in zerochain, so the continuous supply is ensured with some minimal rate, i.e. it can not struck with no offer condition for an indefinite period of time. if the shard is a multisig M2 - each multisig participant has to be such a masternode.