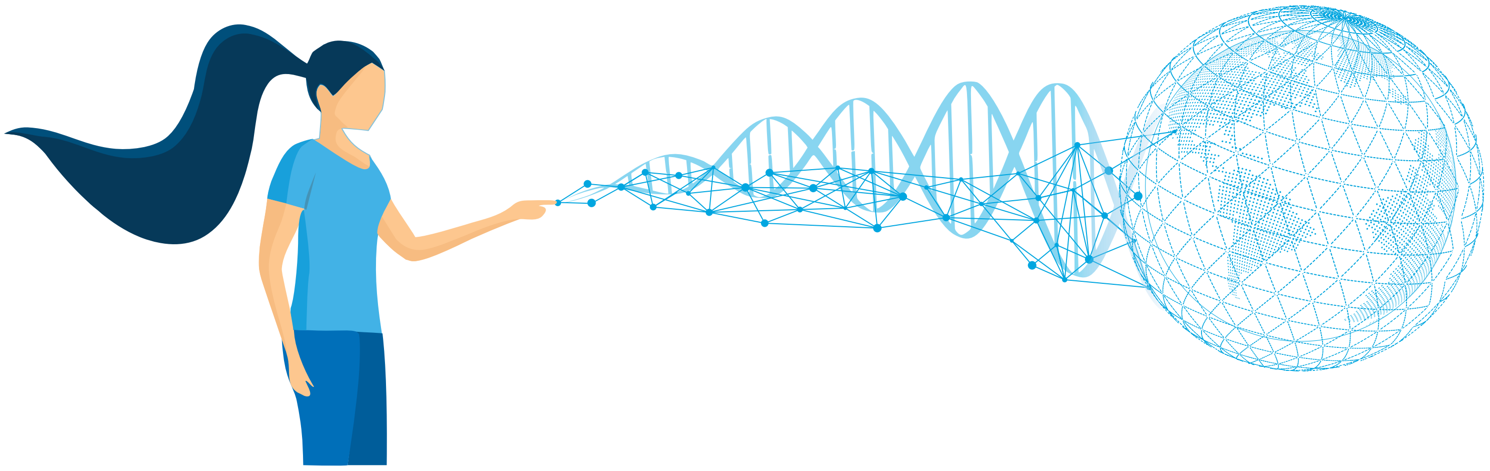
UNIRIS

Be The Only Key

Yellow Paper Season 1

A Truly Decentralized and Limitless Network

14th July 2019



Information Technology has propelled Humanity to the heart of a new paradox. Every individual must own a digital identity to have economic and social interactions (e.g. Network Registration, Platform Logins, Online Payments, App Subscriptions, Chats, File Sharing) ... Without a digital identity, everything is locked. It has become mandatory to exist. However, today, no solution has been found to protect or control this identity. There is no way to be authenticated in an absolutely secure and certain way. There is no way for an individual to prove that they are the one connecting, nor to guarantee that no one else will be able to connect in their place - While, it has never been easier to take control of someone's digital identity (social engineering, zero-days vulnerabilities, etc.).

How will it be possible to guarantee complete trust in an information system without need of trusted third parties? Or, more precisely: *how can we ensure that a system, that we know in advance some will try to hijack*, can guarantee the security of data and information? The principles of Blockchain and in particular the Bitcoin network provided a perfect answer to that issue. However, despite its conceptual perfection and effective robustness demonstrated over the last decade, several weaknesses have emerged:

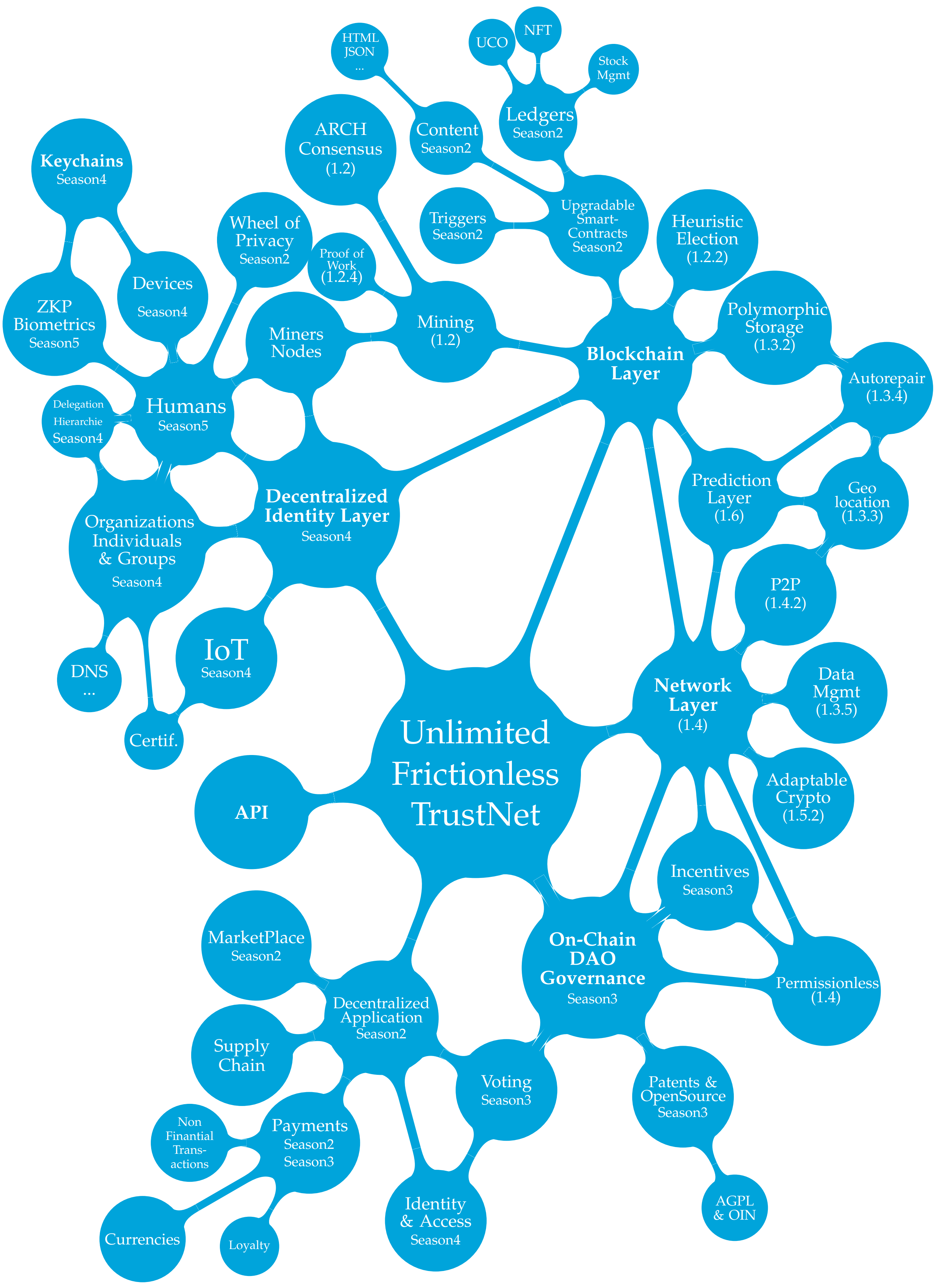
- Complexity for users to secure and safeguard their wallets keys,
- Limited scalability that prevents massive utilization and hence global adoption,
- Too much energy consumption,
- Professionalization and pooling of the miners that has led to a form of recentralization,
- Fratricidal governance that ignores a key actor: *the user*

Current Blockchains do not address the crucial problem of securing identity either: How to guarantee the identity of a user behind a transaction without any disclosure of personal information? How to open your house door, start your car, save your bitcoin keys, pay, vote... etc. without the need for a password or an object (card, badge, smartphone)? *Passwords* get forgotten, copied and hacked while *Objects* get lost, left behind, broken or stolen.

Best solution for user identity remains using the human body itself, in a way it has never been used before. Existing biometric solutions consist of storing and comparing images (Facial recognition, Fingerprint recognition, Retinal recognition), which still carry the risk of breaches similar to the ones mentioned above. To address that challenge Uniris brings a technology that uses the fundamental differentiation between individuals, independent from any personal device. On that basis generate, on the fly, a reproducible/adaptable cryptographic keys that get deleted after use, never stored, enabling individuals to have full control over their identity. This new type of access is provided through a blockchain that takes advantage of the lessons learned from the Bitcoin network and preserves the best of it: "a decentralization capable of proving that the control is not retained by some, but by all".

In the following document and through five successive publications, we present the Uniris solution and how its blockchain provides a counterpoint to every single weakness of current blockchains – provides a decentralized and interoperable identity for every use-case – supports millions of transactions per second avoiding known security breaches – provides a governance integrating the community as a whole and, finally, offers the prospect of a tamper-proof, universal and risk-free human-machine interaction system.

> The Yellow Paper of the Uniris network is divided into 5 Seasons :
>
> Season 1 *A truly decentralized and limitless network*
> Season 2 *Applications anchored in real life*
> Season 3 *Governance and Economic System tailored for humanity*
> Season 4 *Universal and Truly Decentralized Identification*
> Season 5 *Augmented Human*


Unlimited Frictionless TrustNet
API
Blockchain Layer
Mining (1.2)
Proof of Work (1.2.4)
ARCH Consensus (1.2)
Miners Nodes
Upgradable Smart-Contracts Season2
Triggers Season2
Content Season2
HTML JSON ...
Ledgers Season2
UCO
NFT
Stock Mgmt
Heuristic Election (1.2.2)
Polymorphic Storage (1.3.2)
Autorepair (1.3.4)
Prediction Layer (1.6)
Geo location (1.3.3)
P2P (1.4.2)
Network Layer (1.4)
Data Mgmt (1.3.5)
Adaptable Crypto (1.5.2)
Incentives Season3
Permissionless (1.4)
On-Chain DAO Governance Season3
Patents & OpenSource Season3
AGPL & OIN
Voting Season3
Identity & Access Season4
Decentralized Application Season2
MarketPlace Season2
Supply Chain
Payments Season2 Season3
Non Finantial Trans-actions
Currencies
Loyalty
Decentralized Identity Layer Season4
Humans Season5
Wheel of Privacy Season2
Devices Season4
Keychains Season4
ZKP Biometrics Season5
Delegation Hierarchie Season4
Organizations Individuals & Groups Season4
DNS ...
Certif.
IoT Season4

Season 1

A Truly Decentralized & Limitless Network

Given the universal constraints both material and physical, billions of transactions cannot be integrated into a single branch of chained blocks. Similarly, regardless of the consensus method, it is not possible to ensure universal consensus on billions of transactions by polling all nodes of the network. Finally, the functioning of current distributed networks (P2P) is such that it is not possible to guarantee the freshness (consistency) of data on an asynchronous network, unless the network is slowed down excessively by the calculation of the nonce of the block (PoW), as is the case with the Bitcoin network.

Uniris has resolved those 3 issues in the following way:

Transactions Chains: Instead of chained blocks of transactions, each block is reduced to its atomic form, i.e. each block contains only one transaction and each transaction will be chained in its own chain.

ARCH Consensus (Atomic Rotating Commitment Heuristic): This is a new generation of universal consensus allowing atomic commitment by a heuristic rotating election of a tiny set of validation nodes instead of polling the whole network.

Predictive, Optimized, Self-repair Replication System: Instead of synchronizing transactions anarchically on the whole network, every single transaction chain will be stored in a deterministic way and ordered on a set of nodes. Thus, every node will autonomously know all the nodes hosting a given transaction chain and hence ease the network by only interrogating the nearest elected nodes.

A Distributed Network without bottlenecks: Based on Supervised Multicasting, the peer-to-peer network uses a self-discovery mechanism based on incoming connections and the network transactions chains mechanism to maintain a qualified and trusted vision of the network while limiting the generation of new requests on the network.

Season 1 specifically addresses the following topics:

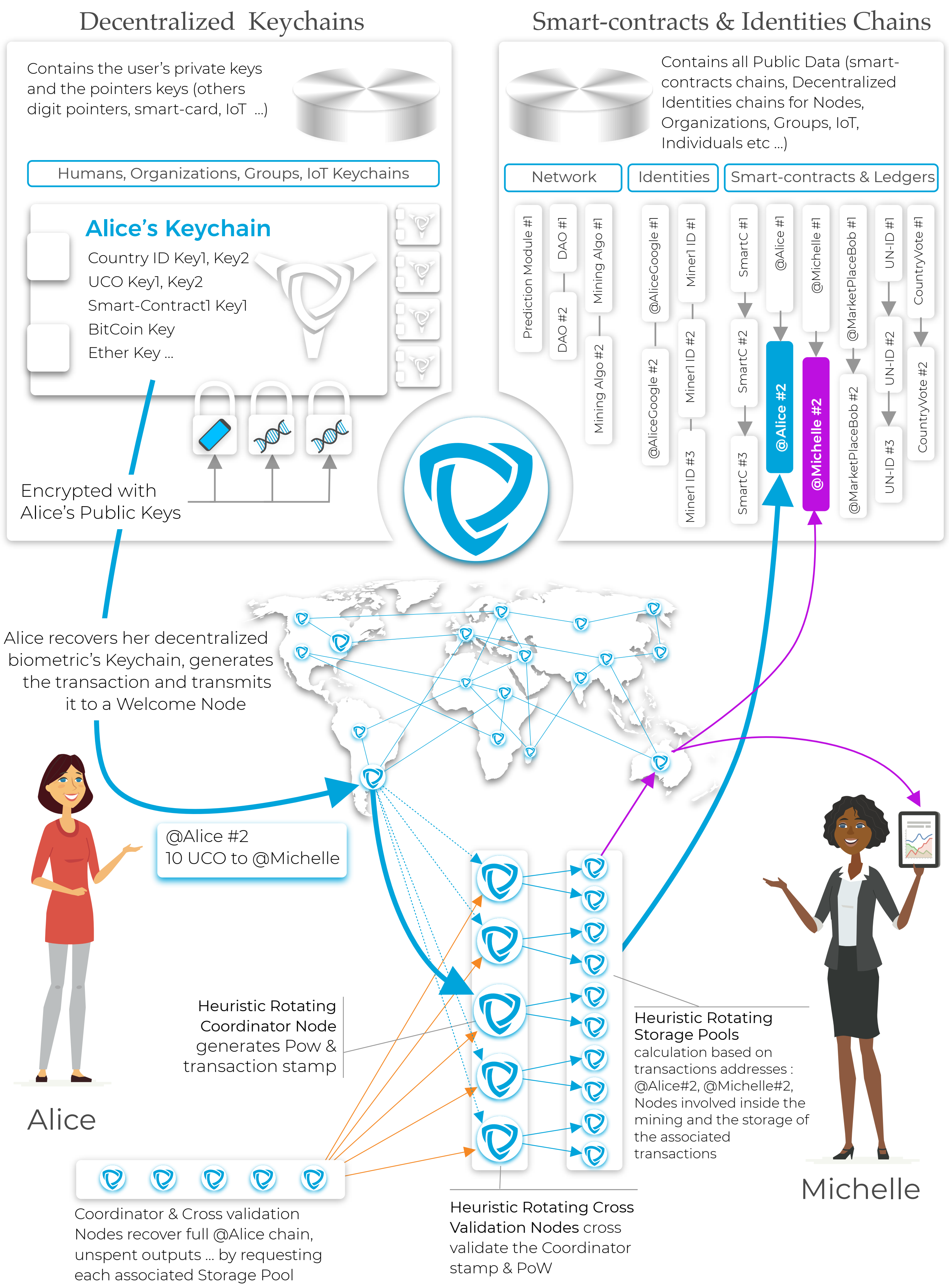
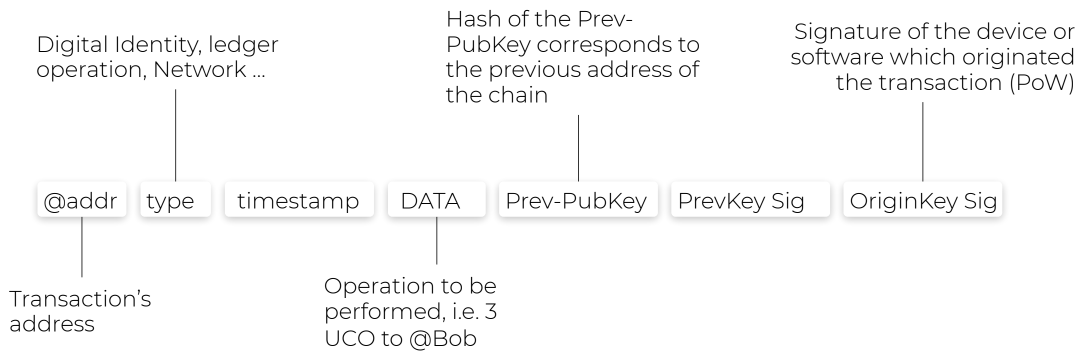

Decentralized Keychains
Contains the user's private keys and the pointers keys (others digit pointers, smart-card, IoT ...)
Humans, Organizations, Groups, IoT Keychains
Alice's Keychain
Country ID Key1, Key2
UCO Key1, Key2
Smart-Contract1 Key1
BitCoin Key
Ether Key ...
Encrypted with Alice's Public Keys
Smart-contracts & Identities Chains
Contains all Public Data (smart-contracts chains, Decentralized Identities chains for Nodes, Organizations, Groups, IoT, Individuals etc ...)
Network
Identities
Smart-contracts & Ledgers
Prediction Module #1
DAO #1
DAO #2
Mining Algo #1
Mining Algo #2
@AliceGoogle #1
@AliceGoogle #2
Miner1 ID #1
Miner1 ID #2
Miner1 ID #3
SmartC #1
SmartC #2
SmartC #3
@Alice #1
@Alice #2
@Michelle #1
@Michelle #2
@MarketPlaceBob #1
@MarketPlaceBob #2
UN-ID #1
UN-ID #2
UN-ID #3
CountryVote #1
CountryVote #2
Alice recovers her decentralized biometric's Keychain, generates the transaction and transmits it to a Welcome Node
@Alice #2
10 UCO to @Michelle
Alice
Michelle
Heuristic Rotating Coordinator Node generates Pow & transaction stamp
Heuristic Rotating Storage Pools calculation based on transactions addresses : @Alice#2, @Michelle#2, Nodes involved inside the mining and the storage of the associated transactions
Coordinator & Cross validation Nodes recover full @Alice chain, unspent outputs ... by requesting each associated Storage Pool
Heuristic Rotating Cross Validation Nodes cross validate the Coordinator stamp & PoW

Figure 1.1: Uniris Chain Overall Functioning

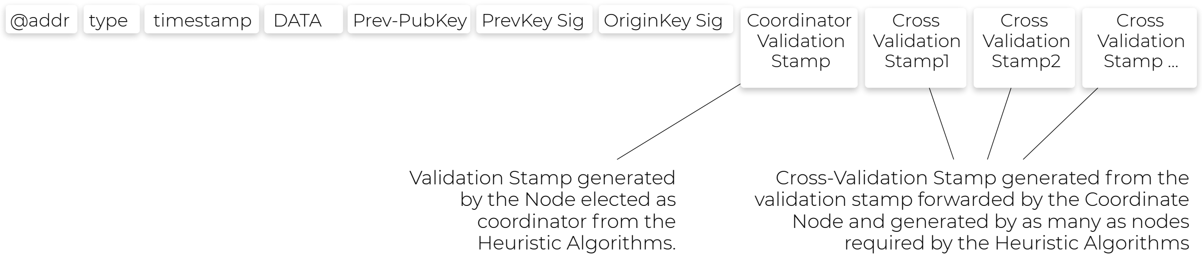
1.1 The Unexplored Eldorado of Transactions Chains

As described above, each block is reduced to its atomic form, i.e. each block is a transaction with its own validation evidences that will allow it to be associated with a given chain. Each transaction also has an additional signature *(Figure 1.2)* corresponding to the signature of the device that generated the transaction. This signature is used inside the Proof of Work mechanism and can be integrated as a necessary condition for the validation of a transaction (for example, in the context of an electronic vote, this will ensure that a person's biometric identity is generated only through a recognized device).



Digital Identity, ledger operation, Network ...
Hash of the Prev-PubKey corresponds to the previous address of the chain
Signature of the device or software which originated the transaction (PoW)
@addr
type
timestamp
DATA
Prev-PubKey
PrevKey Sig
OriginKey Sig
Transaction's address
Operation to be performed, i.e. 3 UCO to @Bob

Figure 1.2: Pending Transaction



@addr
type
timestamp
DATA
Prev-PubKey
PrevKey Sig
OriginKey Sig
Coordinator Validation Stamp
Cross Validation Stamp1
Cross Validation Stamp2
Cross Validation Stamp ...
Validation Stamp generated by the Node elected as coordinator from the Heuristic Algorithms.
Cross-Validation Stamp generated from the validation stamp forwarded by the Coordinate Node and generated by as many as nodes required by the Heuristic Algorithms

Figure 1.3: Validated Transaction

The principles of Transactions Chains are as follows:

Principle 1. Pending Transaction is a transaction that does not have proof of network validation. To be processed by the network, this transaction must have the following information (Figure 1.2) :

@addr	An unused address that corresponds to the hash of the public key of the transaction that will succeed it in the chain.
type	A type defining the functional role of the pending transaction.
timestamp	The date and time of the transaction generation.
DATA	Data zone containing all the operations to be performed (operation on one of the registers, programmed tasks, smart-contracts, operations on an identity, etc.)

Prev-PubKey The public key associated with the previous transaction (the address of the previous transaction being the hash of this public key)

PrevKey Sig Signature from the private key associated with the mentioned public key to prove the possession of the private key, the chaining of transactions and the content of the transaction (address, type, timestamp and data area "DATA").

OriginKey Sig Signature from the private key associated with the device or software from which the pending transaction was generated. This signature is used for the proof of work of the nodes (1.2.4).

Principle 2. Validated Transaction is a pending transaction completed with the validation proofs required by the Heuristic Algorithms. These validation proofs are defined as follows (Figure 1.3):

Validation Stamp The stamp generated by the elected coordinator node and containing:

Proof of integrity proving the linkage of the previous transactions
PoW result of the Proof of Work
Ledgers Operations contains all the ledger operations that will be taken into account by the network:

- the ledger movements mentioned in the pending transaction DATA/Ledgers along with the issuer's signature.
- the list of unspent transaction outputs (UTXO) of the sender's chain on the address of the new transaction.
- the ledger movements to the nodes that participated in the validation and retrieval of the data (effective payment of the welcome, coordinator, cross-validation nodes and nodes from which the data were downloaded) DATA/Ledgers/Fee – the *fees* can be mentioned explicitly or not mentioned (if not, the costs will be calculated by the coordinator node).

Cryptographic signature of the Validation Stamp of the coordinating node, the public key being already mentioned in the list of beneficiaries of the fees.

Cross Validation Stamps To be considered as valid, the *Validation Stamp* must be joined by as many *Cross Validation Stamps* as required by the Heuristics Algorithms. Cross Validation Stamps are the signatures of the Validation Stamp by each of the cross-validation nodes (in case of inconsistency or disagreement it will contain the list of inconsistencies noted, all signed by the cross validation node).

Principle 3. Transactions Chains Each validated transaction is stored as a chain that can only be updated from the last validated transaction in the chain.

Principle 4. Network Transactions Chains Identical to other transactions chains but use different validation and replication algorithms. Network transactions chains are replicated on all nodes.

Principle 5. Heuristic Algorithms Set of algorithms, software and configuration files stored in the form of chains and belonging to the "Network Transactions Chains". This includes, but not limited to, election of validation and storage nodes, updating of node keys, Shared Secrets, Interpreters, Prediction Module, Beacon Chain management, etc.

Principle 6. Principles of Replication After a transaction has been validated, the validation nodes will be in charge of applying the following replication process:

- All transactions associated with the same chain must be replicated on the storage nodes associated with the address of the last validated transaction in the chain. Transactions chains associated with network operation must be fully replicated on all nodes (Node chains, algorithms, decentralized softwares, etc.).
- Each validated transaction must be replicated on the storage nodes associated with the addresses of the associated transactions (input/output).
- Each transaction address must be replicated on the storage nodes of the associated address subset *(Beacon Chains)*.

Principle 7. Principles of Synchronization Each node as soon as it arrives on the network must permanently synchronize the transactions chains for which it has been elected as storage node from the Heuristic Algorithms and from the addresses of the last transactions of the chains, each node must also synchronize the transactions to the chains *(in particular unspent outputs)* as long as they are not integrated in a validation stamp (i.e. as long as they have not been consumed in a new transaction as a spent output *Self Repair and Reconfiguration of the network*).

Principle 8. Node Authentication To participate in the network, each node must be authorized and authenticated through a valid transaction chain. Therefore, as soon as its key has expired or when information necessary for the network is modified (IP, port, protocol), the node must regenerate a new transaction on its chain.

Principle 9. Management of Anomalies by Consensus In case of inconsistency or disagreement on the validity of a "transaction in the process of validation", any elected node associated with the validation or storage process may then submit this "potential anomaly" to a validation pool (via a public election process) which must decide on the validity of the transaction and, if applicable, on the origin of the anomaly (network problem, ambiguity of the transaction or maliciousness characterized).

Principle 10. Management of "Malicious" Nodes If the origin of the anomaly associated with the invalidity of a "transaction in the process of validation" is ruled as "Malicious", then the nodes associated with the validation of a "validated transaction" finally defined as invalid will be banned from the network.

Principle 11. UTXO Model A validated transaction cannot change state (i.e a stateless UTXO model[1]), which means that there is no reality outside the validated transactions – each user can check the validity of previous transactions.

Definition 1. To ensure the atomicity property of the validation of a pending transaction with a margin of error of 10^{-9} and always considering that potentially 90% of the network could be "malicious", the number of "n_v" nodes involved in transaction validation and chain replication must always satisfy the following property:

$$\forall\, n_v \in \mathbb{N},\ \sum_{k=1}^{0.1\times n} \frac{\binom{0.1\times n}{k} \times \binom{0.9\times n}{n_v - k}}{\binom{n}{n_v}} \approx 10^{-9} \tag{1.1}$$

[1]Only an unspent transaction output (UTXO) can be spent/used as an input into a new transaction.

where:

n_v : is the number of nodes involved in the validation ;
n : is the total number of nodes in the network.

A pending transaction must receive unanimous, positive and consistent agreement from all nodes n_v to be considered as validated.

Application: *(Definition 1) For a network with 100,000 nodes (n = 100,000) then for a pending transaction to be validated it will need 197 distributed confirmations between the validation nodes and the storage nodes (n_v=197)*

As a corollary:

Remark 1. (Principle 3) The address of any transaction in the same chain could be used as a destination address, it is not necessary to specify the last transaction in the chain. *(The nodes will automatically replicate the transaction on the storage pool associated with the last transaction in the chain).*

Remark 2. (Principle 3) In terms of security, once the public key is disclosed, it is considered expired, only the hash of the public key of the next transaction $\boxed{\text{@addr}}$ is announced, which allows to keep the public key secret until the next transaction on this same chain (thus correcting and combating the problem of repeated use of the same ECDSA key) .

Remark 3. (Principle 3 – 6) When updating a transaction chain through a new validated transaction at the end of the chain. The nodes in charge of storing the chain before the arrival of the new transaction will then be able to stop their synchronization of the chain's data. The new storage nodes now being the referenced storage nodes elected from the address of this last transaction (see 1.4.1 – 1.10).

Remark 4. (Principles 3 – 6) In terms of sequencing, this means that all outputs of a given chain must be serialized (Alice to {Bob, Tom, etc.} then later, Alice to {Michelle} etc.), but an unlimited number of transactions can be sent to the same transaction chain (Bob to {Alice} in the same time than Tom to {Alice}, Michelle to {Alice}, etc.) – This means that a new transaction on a specific transaction chain must wait until the end of the validation of the previous one (1 to 3 seconds), but an almost unlimited number of transactions/second can be sent at the same time to the same chain.

Remark 5. (Principle 2 – 11) The list of unspent outputs does not need to be specified by the sender of the transaction, all unspent or unused outputs are reintegrated directly into the last transaction in the sender's chain and by the Coordinator Node inside the Validation Stamp - allowing nodes to directly find all unspent outputs on the storage nodes of the last transaction and allowing the user to avoid the costly search phase of unspent outputs.

Remark 6. (Principle 11) Like the Bitcoin network protocol, the Uniris network uses the Principle of the UTXO model for the sake of security and non-repudiation of transactions, which means that there is no reality outside the validated transactions. Thus, unlike smart-contracts such as those running inside virtual machines (Ethereum, EOS, etc.), Uniris smart-contracts cannot change their state once validated. This is made possible without reducing the ability to handle numerous disparate use cases (see. Season 2 : Smart-Contracts).

Remark 7. (Principles 3, 4 and 8) The figure 1.4 below represents four types of chained transactions (a chain associated with the identity of a network node, a chain associated with the identity of a person, a chain associated with a UCO wallet (Uniris Coin) and a smart contract chain). The first transaction in the chain (i.e genesis) is widely used as the reference for the chain and the last transaction is the only one that can be updated by a new transaction (the address "@node0-3" announcing the hash of the public key that will allow to verify the possession of the private key, thus allowing the chain to be updated).

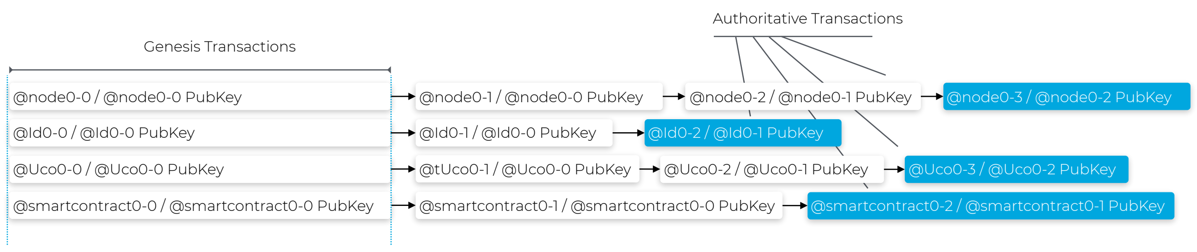
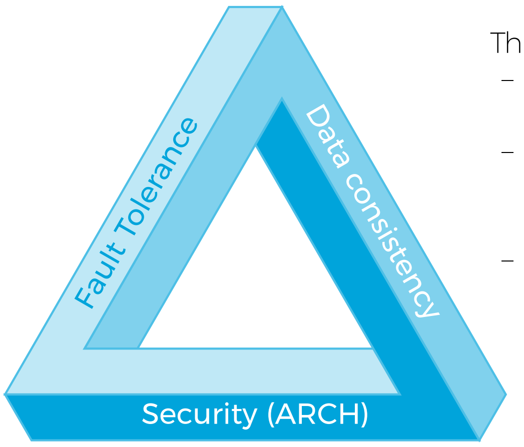

Authoritative Transactions
Genesis Transactions
@node0-0 / @node0-0 PubKey
@node0-1 / @node0-0 PubKey
@node0-2 / @node0-1 PubKey
@node0-3 / @node0-2 PubKey
@Id0-0 / @Id0-0 PubKey
@Id0-1 / @Id0-0 PubKey
@Id0-2 / @Id0-1 PubKey
@Uco0-0 / @Uco0-0 PubKey
@tUco0-1 / @Uco0-0 PubKey
@Uco0-2 / @Uco0-1 PubKey
@Uco0-3 / @Uco0-2 PubKey
@smartcontract0-0 / @smartcontract0-0 PubKey
@smartcontract0-1 / @smartcontract0-0 PubKey
@smartcontract0-2 / @smartcontract0-1 PubKey

Figure 1.4: Examples of transaction chains

1.2 ARCH : An Uncompromising Consensus allowing to reach the million txn/sec



The Uniris network is based on three properties:

- Security: based on the ARCH consensus, each transaction is validated atomically.
- Data consistency: based on heuristic storage algorithms that guarantee access to the latest write but also maximum data availability.
- Fault tolerance: based on heuristic validation algorithms that allow nodes to work independently even in the event of a network disaster.

ARCH Consensus or "Atomic Rotating Commitment Heuristic (ARCH)" is a new generation of Consensus. Below is a detailed explanation of each concept of ARCH:

Atomic Commitment is the form of "absolute" consensus that implies 100% concordant responses for the acceptance or refusal of the transaction validation

Heuristics is the set of algorithms, softwares and parameters that manage the entire network, allowing the network to elect in a decentralized and coordinated way the nodes in charge of validating and storing transactions chains.

Rotating the network being fully distributed (no central or privileged role), the nodes elected for each operation are constantly changing so that no node can predict which nodes will be elected until the transaction arrives.

The Uniris network is based on hypergeometric distribution laws which, from an unpredictable election and a formal consensus, make it possible to obtain with certainty (99.9999999%) the same answer by querying 197 nodes as would be obtained by querying 100,000 (*Network Resilience & Malicious Operations Detection [1.5.1]*). In other words, this mathematical law makes it possible to obtain an universal consensus from a small part of the nodes - this property thus enters into the *Heuristics* concept widely used on the whole network. The risk of the related availability is ensured by a strict management of the disruptive nodes *(Principles 8, 10 and 11)*, which are banished after investigation of the origin of the disagreement.

These concepts are defined in the following sections:

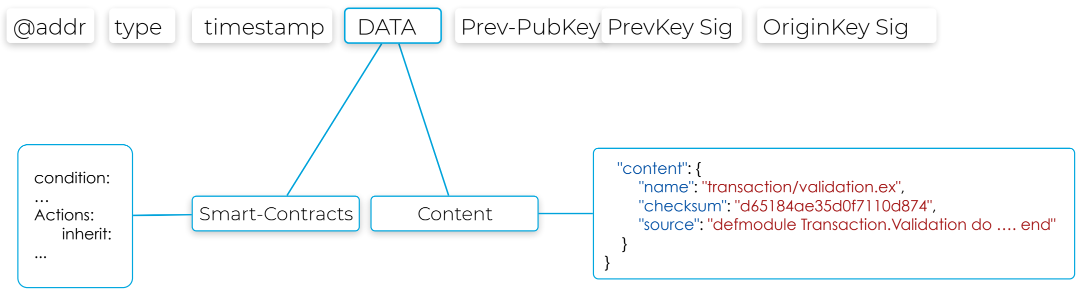
1.2.1 Heuristic Algorithms

Heuristic Algorithms are made up of softwares (interpreters, libraries, etc.) and configuration files *(Principle 5)*.

To provide a trusted and fully decentralized network, there is the problem of trust on the software and hence the algorithmic layers of the network. Indeed, how to achieve atomic validation of a transaction if the nodes do not use the same algorithms and interpreters and hence, how to differentiate a malicious node from a simply obsolete node. Similarly, it is not possible to ensure the reality of decentralized governance if the nodes are autonomous in choosing the software/algorithmic layer they use.

To solve these problems, Heuristic Algorithms/Softwares are stored in a decentralized way as transaction chains or more precisely as smart-contract chains. The governance and operation of the associated smart-contracts being fully configurable: Self-Triggable, protected by Inherited Recursive Constraints *(Season 2)* and under the control of the community *(Season 3)*.

The example in the figure 1.5 below represents a module stored with its source code in the DATA/Content area and containing Inherited Recursive Constraints



```
"content": {
    "name": "transaction/validation.ex",
    "checksum": "d65184ae35d0f7110d874",
    "source": "defmodule Transaction.Validation do .... end"
  }
}
```

Figure 1.5: Decentralized software version management and Heuristic Algorithms

1.2.2 Global, Unpredictable and Reproducible Election

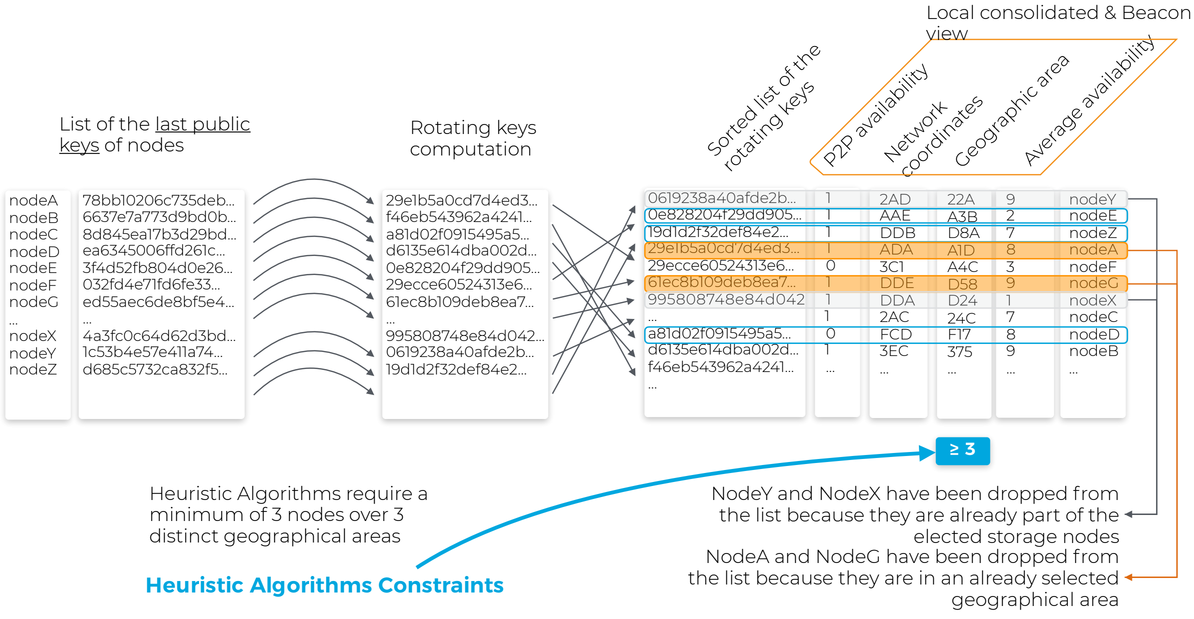
To achieve an unpredictable, global but locally executed, verifiable and reproducible election of transaction validation nodes, the Heuristics Algorithms are based on

- an unpredictable element: the hash of the transaction content,
- an element known only by the authorised nodes: the *Daily Nonce* included in the Shared Secrets of the nodes and renewed daily *(Figure 1.19)*,
- an element difficult to predict: the last public key of the node chain: *Last Node Pubkey*,
- the computation of the *Validation Rotating Keys*

This computation works as follow:

$$\text{RotatingKey}_{NodePubkey} = \text{Hash}\big(\textit{Last Node Pubkey, Daily Nonce, hash(transaction content)}\big) \tag{1.2}$$

The figure 1.6, below, schematically represents how the algorithms work to get a filtered list of validation nodes (i.e Validation Pool):



Local consolidated & Beacon view
List of the last public keys of nodes
Rotating keys computation
Sorted list of the rotating keys
P2P availability
Network coordinates
Geographic area
Average availability
≥ 3
Heuristic Algorithms Constraints

nodeA	78bb10206c735deb...
nodeB	6637e7a773d9bd0b...
nodeC	8d845ea17b3d29bd...
nodeD	ea6345006ffd261c...
nodeE	3f4d52fb804d0e26...
nodeF	032fd4e71fd6fe33...
nodeG	ed55aec6de8bf5e4...
...	...
nodeX	4a3fc0c64d62d3bd...
nodeY	1c53b4e57e411a74...
nodeZ	d685c5732ca832f5...

29e1b5a0cd7d4ed3...
f46eb543962a4241...
a81d02f0915495a5...
d6135e614dba002d...
0e828204f29dd905...
29ecce60524313e6...
61ec8b109deb8ea7...
...
995808748e84d042...
0619238a40afde2b...
19d1d2f32def84e2...

Sorted list of the rotating keys	P2P availability	Network coordinates	Geographic area	Average availability	
0619238a40afde2b...	1	2AD	22A	9	nodeY
0e828204f29dd905...	1	AAE	A3B	2	nodeE
19d1d2f32def84e2...	1	DDB	D8A	7	nodeZ
29e1b5a0cd7d4ed3...	1	ADA	A1D	8	nodeA
29ecce60524313e6...	0	3C1	A4C	3	nodeF
61ec8b109deb8ea7...	1	DDE	D58	9	nodeG
995808748e84d042	1	DDA	D24	1	nodeX
...	1	2AC	24C	7	nodeC
a81d02f0915495a5...	0	FCD	F17	8	nodeD
d6135e614dba002d...	1	3EC	375	9	nodeB
f46eb543962a4241...	...	...	...	...	...
...					

Heuristic Algorithms require a minimum of 3 nodes over 3 distinct geographical areas

NodeY and NodeX have been dropped from the list because they are already part of the elected storage nodes

NodeA and NodeG have been dropped from the list because they are in an already selected geographical area

Figure 1.6: Election of Mining Validation Nodes

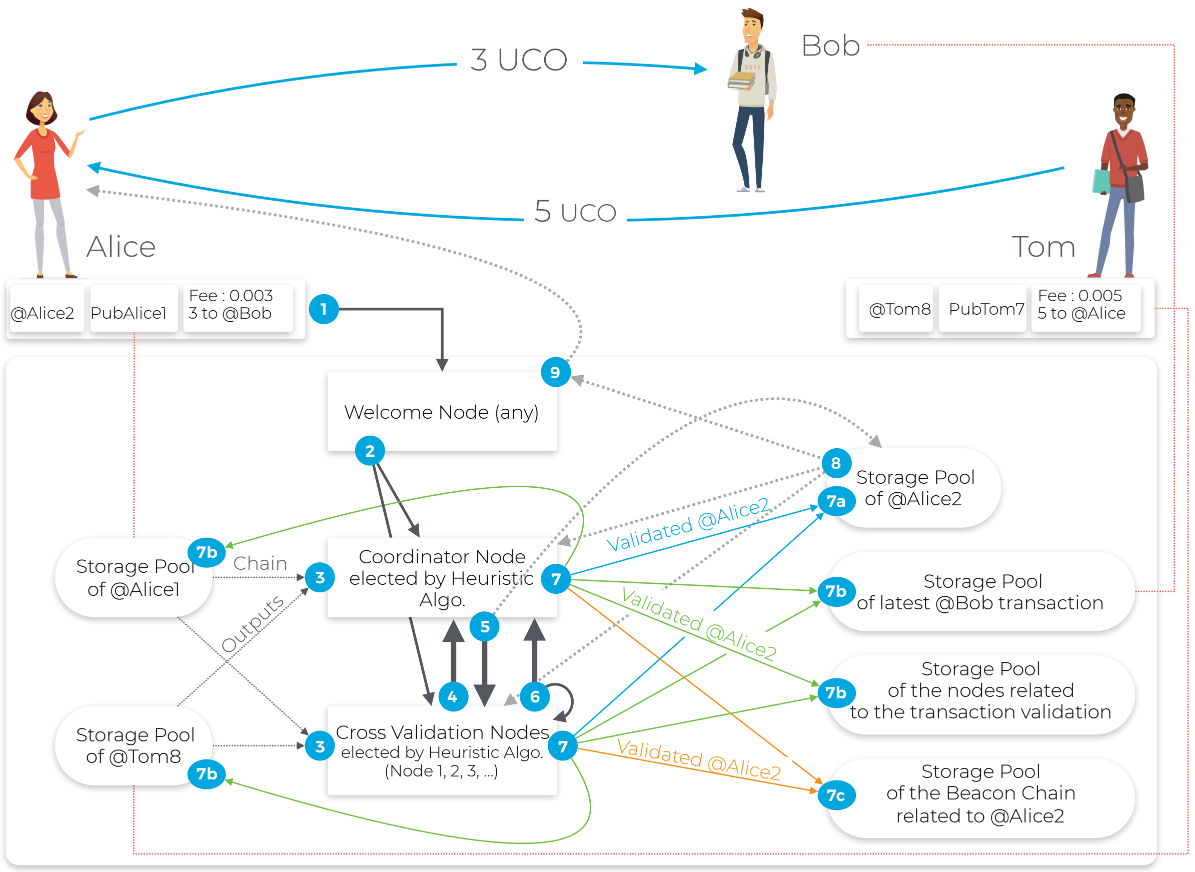
The purpose of the rotating keys calculation is to provide an unpredictable and reproducible ordered list of the allowed nodes. The scheduling thus obtained allows each of the nodes to find autonomously and share the list of nodes that will be in charge of the validation of the transaction. This list is then filtered according to the constraints of the Heuristic Algorithms from the consolidated views of the nodes consisting of:

- The view of the local network of the node*(Figure 1.17)*
- Update of replicated network chains across all nodes *(Principle 5)*
- Information provided by beacon chains *(Figure 1.15)*
- Constraints added by the prediction module *(Figure 1.21)*

Since this validation is a one-time process, the priority criteria taken into account are the availability of the node at the time of the election, the geographical area, the number of nodes to be elected and the list of storage nodes that will participate in a second phase to the validation of the transaction (1.3.2).

1.2.3 Transaction validation process

The diagram below illustrates the validation process of a UCO (Uniris Coin) transfer transaction in a simplified way:



3 UCO
Bob
5 UCO
Alice
Tom
@Alice2
PubAlice1
Fee : 0.003
3 to @Bob
@Tom8
PubTom7
Fee : 0.005
5 to @Alice
1
9
Welcome Node (any)
2
8
Storage Pool of @Alice2
7a
Validated @Alice2
Storage Pool of @Alice1
7b
Chain
3
Coordinator Node elected by Heuristic Algo.
7
7b
Storage Pool of latest @Bob transaction
Validated @Alice2
Outputs
5
Storage Pool of the nodes related to the transaction validation
4
6
Storage Pool of @Tom8
3
Cross Validation Nodes elected by Heuristic Algo. (Node 1, 2, 3, ...)
7
Validated @Alice2
7c
Storage Pool of the Beacon Chain related to @Alice2

Figure 1.7: Transaction validation process

1 Alice sends the @Alice2 transaction to any node in the network (welcome node) through the "Seeds" list (1.4.3)
2 The welcome node determines the validation nodes (coordinator and cross-validation) using Heuristic Algorithms (1.2.2), then forwards the transaction to the chosen nodes so that they can start the pre-validation work.
3 The elected nodes (coordinators and cross validation nodes) get all the transactions necessary for validation from the closest storage nodes (1.3.3) : the complete chain on the storage nodes of the transaction @Alice1 (1.3.2), and all transactions related to unspent entries (UTXO) @Tom8... on the respective storage pools.
4 Once the context of the transaction has been rebuilt, the cross-validation nodes communicate to the coordinating node the list of storage nodes used to collect the data and their views on the availability of the validation and storage nodes.
5 The Coordinator node, later has to rebuild the context of the transaction (chain, inputs, output), figure out the proof of work (PoW), compute the replication tree , define the operations on the ledgers and sign the validation stamp *(Validation Stamp – Principle 2)* and transmit it to the cross validation nodes.
6 The content of the Validation Stamp is verified by each cross-validation node which will further issue a "Cross-Validation Stamp" to the coordinator and other cross-validation nodes (Principle 2).
7 Once all the Cross-Validation Stamps are received by each validation node, they can start the replication process of the validated transaction, as defined by the Coordinator node:

7a the storage pool of the @Alice2 chain will recover the context of the transaction, if everything is compliant, store it in its local database and 8 notify the validation and welcome nodes of the completeness of the replication.
7b The storage pools assigned to @Bob and the nodes involved will check the transaction's compliance and integrate it into their db.
7c The storage pool related to the Beacon Chain (1.4.1) will check the compliance of the transaction and integrate the timestamp and the addresses involved @Alice1, @Alice2, @Bob into the Beacon chain.

9 The welcome node will notify the user of the transaction validation progress

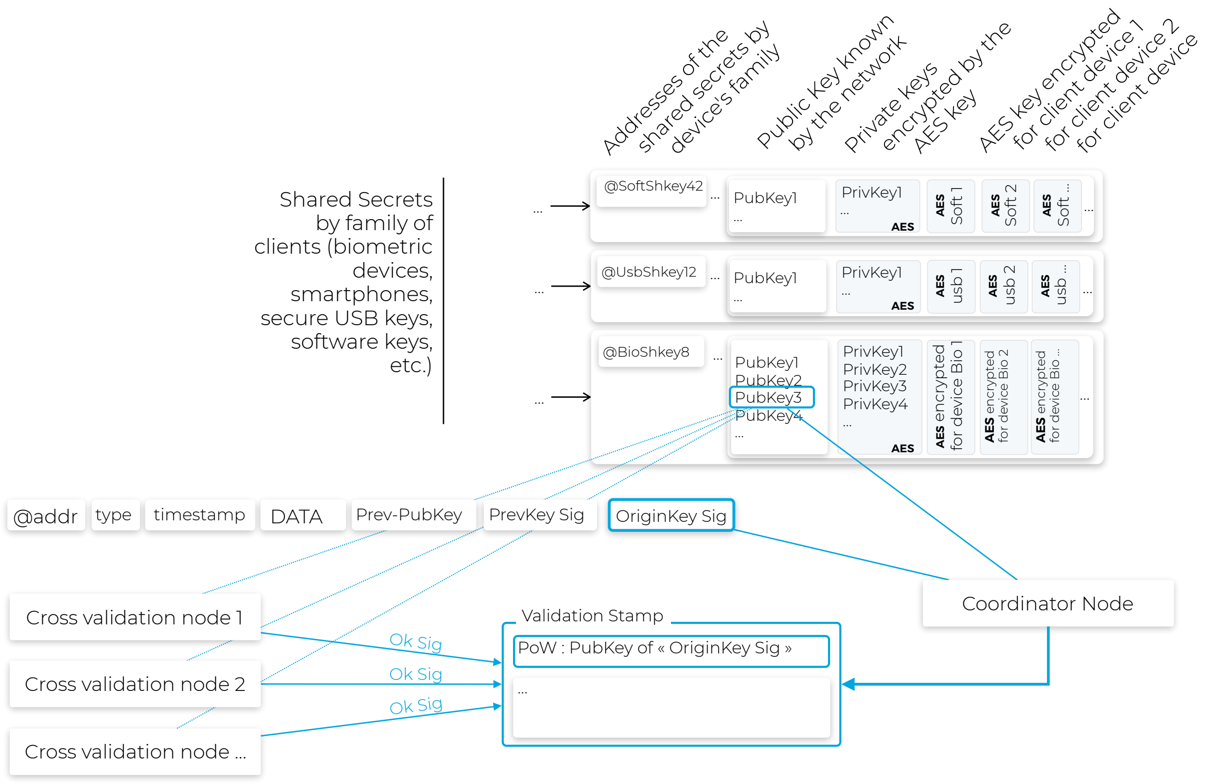
1.2.4 Proof of Work

The Bitcoin and networks based on the principle of Proof of Work (PoW) ensure an unpredictable and pseudo-random election of the nodes in charge of the validation of the blocks (mining). They also ensure that the network remains synchronous and that each node will have time to recover the previous blocks, thanks to the time required to calculate this proof of work (10min).

On the Uniris network:

- The election of the unpredictable and pseudo-random validation nodes is ensured by the Heuristic Validation Algorithms *(Figure 1.6)*.
- The synchronization of the data is ensured by the Heuristic Algorithms for replication of data, which provide the knowledge of the nodes responsible for storing a given transaction chain, at all times *(Figure 1.10)*.
- The proof of work mechanism is nevertheless maintained to ensure the authentication of the transaction origination devices. This allows additional security requirements on transaction validation like: prohibit any transaction even in case of key theft (Season 4) – allow user to consult his balance on any smartphone while only other one allows to generate a transaction – enable the organizers of an election to ensure the actual biometric identity of a voter by the associated cryptographic identification of a particular device.

The *Proof of Work* consists of finding the right public key associated with the signature of the device that originated the OriginKey Sig transaction. This involves a search on a finite set of public keys ensuring that the device is definitely authorized to generate the transaction and that, at the same time, the nodes able to find that key are authorized and know the Shared Secrets of the Nodes *(Figure 1.19)* – otherwise, brute force search may take a considerable amount of time.



Addresses of the shared secrets by device's family
Public Key known by the network
Private keys encrypted by the AES key
AES key encrypted for client device 1 for client device 2 for client device
Shared Secrets by family of clients (biometric devices, smartphones, secure USB keys, software keys, etc.)
@SoftShkey42
PubKey1
PrivKey1
AES
AES Soft 1
AES Soft 2
AES Soft ...
@UsbShkey12
AES usb 1
AES usb 2
AES usb ...
@BioShkey8
PubKey2
PubKey3
PubKey4
PrivKey2
PrivKey3
PrivKey4
AES encrypted for device Bio 1
AES encrypted for device Bio 2
AES encrypted for device Bio ...
@addr
type
timestamp
DATA
Prev-PubKey
PrevKey Sig
OriginKey Sig
Coordinator Node
Cross validation node 1
Cross validation node 2
Cross validation node ...
Ok Sig
Ok Sig
Ok Sig
Validation Stamp
PoW : PubKey of « OriginKey Sig »

Figure 1.8: Proof of Work

This mechanism makes it possible to support an unlimited number of devices while guaranteeing the confidentiality of the user with respect to the device used, the private key used by a device being the last key shared between the devices of the same family (software keys, smartphone keys, smart card keys, etc.). Just like the nodes of the network, each device will have its own transaction chain allowing it to update its keys.

Since the mechanism for renewing the shared keys of the devices is same as the renewal of the shared keys of the nodes (Figure 1.19) the network can also ban a particular device like it can ban a node (the shared key of the devices will no longer be encrypted for the device banned).

1.2.5 Protection against Double, and also Multiple Spending

The potential problem of *double spending* is unique to digital currencies based on decentralized networks. Beyond the financial issues, it poses the problem of operational synchronization of autonomous nodes in lack of any central server scheduling the operations. Each node must therefore be able to locally and autonomously ensure the coherence of the network.

To solve the problem of network desynchronization (non-fraudulent cases), the Uniris network uses:

- The mechanism of transactions chains *(Principle 3)* which allows to serialize the generation of a new transaction on a chain - the generation of a transaction is therefore always synchronous with its chain. Especially for chains using the key derivation mechanism, for which a fork could not technically be stored and must be resolved before storage.
- The Heuristic Storage Algorithms that allow you to know locally, at any time, the order of replication and the order of freshness of each transaction from its address *(Figure 1.10)*. This mechanism, which is used for both transaction chains and for the storage of algorithms and software *(Figure 1.5)* ensures that each node has the necessary means to use synchronous data autonomously.
- Finally, the validation mechanism *(Figure 1.7)* uses cross-notifications from the storage pools to schedule and notify the validation pool in case of double spending. Since the validation pool retrieves the chain and the unspent outputs from the associated storage pool before validation and notifies the associated storage pool after validation.

To solve the problem of double fraudulent double spending (attempted fraud organized by the validation pool), the Uniris network is based on:

- The property of hypergeometric distribution *(Figure 1.18)* to guarantee that even with 90% of malicious nodes, the risk that an honest node cannot detect a fraudulent transaction is only 10^{-9} or one chance in one billion *(Definition 1)*.
- If a fraud case is detected by any replication node *(Principle 9)*, then a public and decentralized investigation process will be initiated to rule on the fraudulent nature of the validated transaction. At the end of this investigation, nodes considered as malicious will be banned from the network *(Principle 10)*.

Beyond the problem of double spending, the transaction chain mechanism also avoids the problem of *multiple simultaneous payments* - so if something disrupts the payment process - a user can send an unlimited number of attempts (using the same previous transaction address) and can be certain that only one transaction (payment) will be considered.

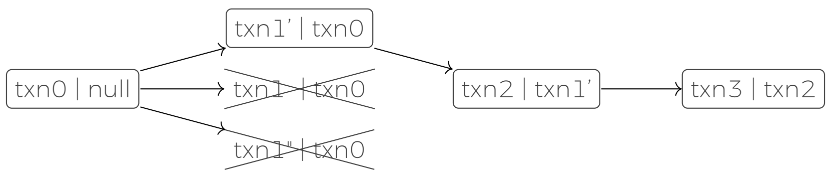


Figure 1.9: Double spending management

1.3 Storage Capacity Increase by tenfold & Geographically Secure

Regardless of the distributed network technology, challenges are:
- How to find a specific data without interrogating the entire network?
- How to make sure that the data is the most up-to-date?
- How to guarantee that data will not be lost or corrupted?
- How to ensure data consistency within a fragmented network?
- How to avoid an attack if the nodes in charge of data are known?
- How to provide an unlimited network without creating an overload of the global network?

To address these constraints, the layers of self-discovery and data synchronization have been completely redesigned to ensure the availability of data regardless of any disaster, to give priority in terms of data freshness and to allow data recovery through the best network path.

In the following sections, we will define these concepts sequentially:

1.3.1 Number of Replicas per Chain

The distributed storage layer is an essential component to allow the network to always store more data. In the Uniris network, data is stored in a sharding manner, i.e. none of the nodes contains all the data and the network will therefore be able to store a linearly larger amount of data as the number of nodes increases.

With the exception of Network Transaction Chains which are stored on all nodes *(Principle 4)*, the rule of the number of replicas per chain is intended to be refined over time through the prediction module *(Figure 1.21)*. As a first approximation, it should correspond to the following formula:

$$\forall n_r \in \mathbb{N}, \sum_{k=1}^{n} \mathit{availability}(k) > 2^{(\log_{10} n + 5)} \tag{1.3}$$

where:

n_r :	Number of replicas required;
n :	Total number of node in the network;
availability :	Average availability of a node on the network.

Unlimited Network Paradigm: The number of replicas per transaction chain (from the address of the last transaction of the chain) will be based:

1. In first implementation on the formula above (1.3)
2. Then reduce the number of replicas based on the hypergeometric distribution ≈ 200
3. Finally, after three validated transactions, keep only the chain condensates (keeping only the cryptographic evidence and movements on the ledgers for the oldest parts). The migration to condensates makes it possible to gain a minimum factor of 10 on the size of the stored chains.

The result in terms of useful storage capacity and considering that the nodes have 512Gb of useful storage gives the following results for each of the above three scenarios:

	1000 nodes	10,000	100,000	1,000,000	Number of transactions			
1	2 Tb	10 Tb	50 Tb	250 Tb	6.10^9	32.10^9	161.10^9	806.10^9
2	2.56 Tb	25.6 Tb	256 Tb	2.56 Pb	8.10^9	82.10^9	825.10^9	8.10^{12}
3	2.56 Tb	25.6 Tb	256 Tb	2.56 Pb	82.10^9	8.10^{11}	8.10^{12}	82.10^{12}

Without data fragmentation (sharding) the maximum allowable size (useful capacity) is limited to the size of the smallest node disk, for example, with 512Gb:

- Bitcoin network can store 830 million transactions (431 million for 266 Gb)
- Ethereum network can store 122 million transactions (489 million ≈ 2 Tb)

For an equivalent number of nodes (100,000), Uniris network can store at least *(Step 1)* 50Tb or 161.10^9 transactions and at most *(Step 3)* 256Tb or 8.10^{12} txn, which is:

- Between 193.9 and 9638 times the number of transaction in the Bitcoin network.
- Between 1319 and 65,573 times the number of transaction in the Ethereum network.

1.3.2 Storage Nodes Election

Unlike the one-time process of election of validation nodes, the election of storage nodes associated with a transaction chain is performed each time the network is modified (ex. each time a node disconnected or join the network) and each time a new transaction is performed on the network. This election computation performed in a few milliseconds, allows each node to autonomously know whether or not to download and store a transaction chain or to no longer store it *(Figure 1.13)*.

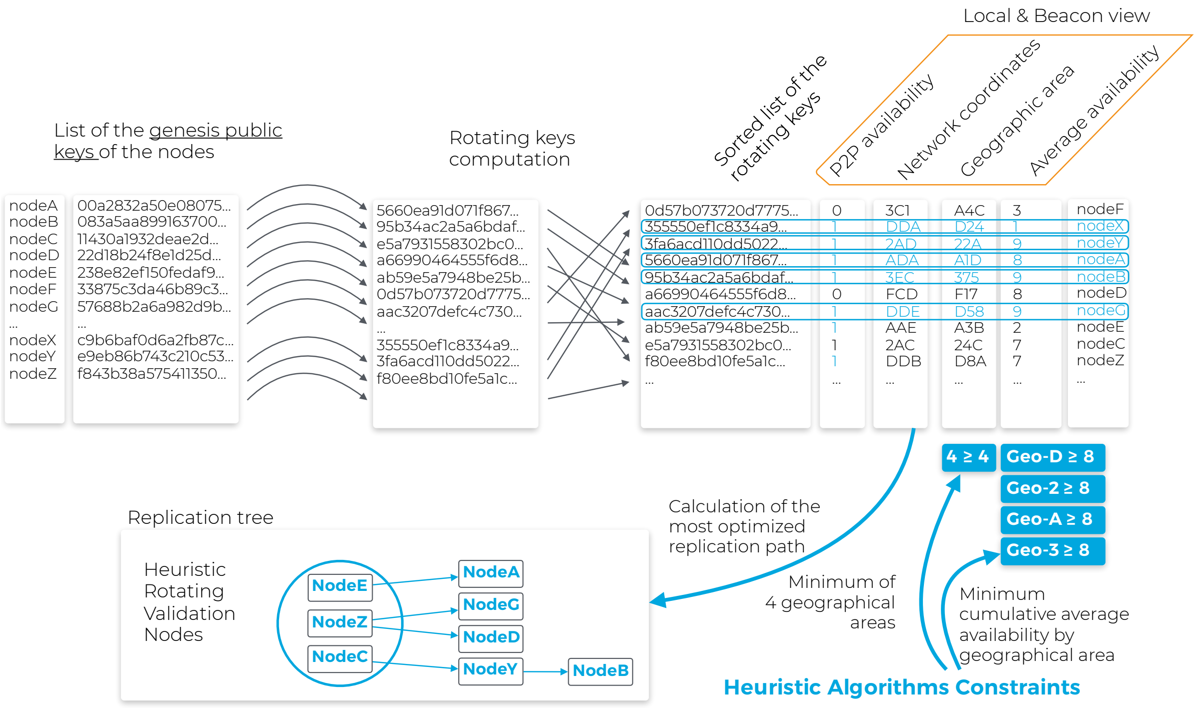
To perform this election, the Rotating Heuristic Storage Algorithm is based:

- On the transaction's address (the transactions chains being stored on the address associated with the last transaction in the chain – *Principle 8*),
- On a stable element known only by the authorized nodes: the *Storage Nonce* included in the Shared Secrets of the nodes *(Figure 1.19)*,
- On the first public key (genesis) of each node's chain: *Genesis Node Pubkey*, the first public key of the nodes being stable, the storage nodes of a given transaction chain will remain constant on a constant network,
- And on the calculation of rotating storage keys.

The computation of the *Storage Rotating Keys* works as follows:

$$Rotating_{Pubkey} = \text{Hash}\left(\textit{Genesis Node Pubkey, Storage Nonce, hash(transaction address)}\right) \qquad (1.4)$$

The figure 1.10 below shows schematically how the algorithms work to obtain a filtered and ordered list of storage nodes (Storage Pool):



List of the genesis public keys of the nodes
nodeA 00a2832a50e08075...
nodeB 083a5aa899163700...
nodeC 11430a1932deae2d...
nodeD 22d18b24f8e1d25d...
nodeE 238e82ef150fedaf9...
nodeF 33875c3da46b89c3...
nodeG 57688b2a6a982d9b...
...
nodeX c9b6baf0d6a2fb87c...
nodeY e9eb86b743c210c53...
nodeZ f843b38a575411350...
Rotating keys computation
5660ea91d071f867...
95b34ac2a5a6bdaf...
e5a7931558302bc0...
a66990464555f6d8...
ab59e5a7948be25b...
0d57b073720d7775...
aac3207defc4c730...
...
355550ef1c8334a9...
3fa6acd110dd5022...
f80ee8bd10fe5a1c...
Sorted list of the rotating keys
Local & Beacon view
P2P availability
Network coordinates
Geographic area
Average availability
0d57b073720d7775... 0 3C1 A4C 3 nodeF
355550ef1c8334a9... 1 DDA D24 1 nodeX
3fa6acd110dd5022... 1 2AD 22A 9 nodeY
5660ea91d071f867... 1 ADA A1D 8 nodeA
95b34ac2a5a6bdaf... 1 3EC 375 9 nodeB
a66990464555f6d8... 0 FCD F17 8 nodeD
aac3207defc4c730... 1 DDE D58 9 nodeG
ab59e5a7948be25b... 1 AAE A3B 2 nodeE
e5a7931558302bc0... 1 2AC 24C 7 nodeC
f80ee8bd10fe5a1c... 1 DDB D8A 7 nodeZ
4 ≥ 4
Geo-D ≥ 8
Geo-2 ≥ 8
Geo-A ≥ 8
Geo-3 ≥ 8
Replication tree
Heuristic Rotating Validation Nodes
NodeE
NodeZ
NodeC
NodeA
NodeG
NodeD
NodeY
NodeB
Calculation of the most optimized replication path
Minimum of 4 geographical areas
Minimum cumulative average availability by geographical area
Heuristic Algorithms Constraints

Figure 1.10: Storage node election process

The objective of the storage rotating key algorithm is to allow nodes to obtain in an autonomous and common way, an ordered and reproducible list of storage nodes from any transaction's address.

To ensure maximum data availability and security of data, this list is refined based on the constraints imposed by the Heuristic Storage Algorithms from the consolidated views of the nodes consisting of:

- The local network view of the node *(Figure 1.17)*
- Update of network chains replicated on all nodes *(Principle 5)*
- Information provided by beacon chains *(Figure 1.15)*
- Constraints added by the prediction module *(Figure 1.21)*

The main parameters considered are:

- Geographical location: allowing continuity of service even in the event of natural disasters in one or more geographical areas. To make the geographical position of a node more reliable, it is contextualized by the network coordinates calculated globally by the beacon chain mechanism *(Figure 1.16)*, and also when renewing the keys associated with the nodes.
- The average availability of nodes in a given area: rather than modifying the list of elected nodes, as in the case of the election of validation nodes, the election of a storage node will be weighted by its availability. The goal will therefore be, the cumulative availability (each geographical area requiring a minimum cumulative availability D = 1+9 > 8...).
- The risk identified by the prediction module on one or more groups of nodes to modify the weightage of node availability.

The reality of a geographical position does not necessarily indicate proximity from the network point of view (latency, throughput). So the replication tree is calculated globally from the local and qualified views of the nodes, at the time of the generation of the validation stamps *Coordinator Validation & Cross Validation Stamps* thus allowing to distribute the replication work from the most optimized paths.

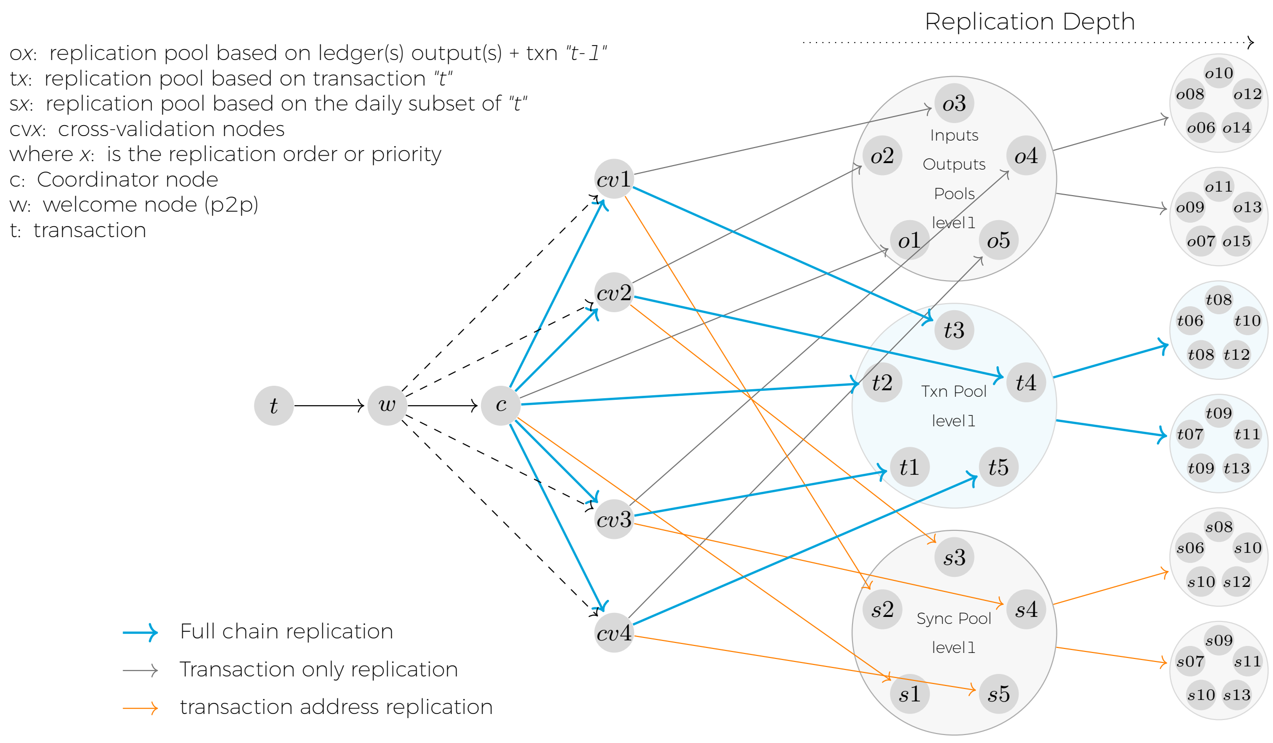

Replication Depth
ox: replication pool based on ledger(s) output(s) + txn "t-1"
tx: replication pool based on transaction "t"
sx: replication pool based on the daily subset of "t"
cvx: cross-validation nodes
where x: is the replication order or priority
c: Coordinator node
w: welcome node (p2p)
t: transaction
t
w
c
cv1
cv2
cv3
cv4
o1
o2
o3
o4
o5
Inputs
Outputs
Pools
level1
t1
t2
t3
t4
t5
Txn Pool
level1
s1
s2
s3
s4
s5
Sync Pool
level1
o10
o08
o12
o06
o14
o11
o09
o13
o07
o15
t08
t06
t10
t08
t12
t09
t07
t11
t09
t13
s08
s06
s10
s10
s12
s09
s07
s11
s10
s13
Full chain replication
Transaction only replication
transaction address replication

Figure 1.11: Functional view of the replication process of a validated transaction

As shown in the figure 1.11 above, once the transaction is validated, the validation nodes will start the replication process from the defined replication tree. To avoid failed replicas, each node will wait for acknowledgement of the next level of replication before stopping its replication process. Similarly, if a node within this replication does not acknowledge, the node at the top level will support this unsuccessful replication to ensure the continuity of the replication.

Since each node has a limited trust in the others *(Banishment risk – Principle 10)*, replication will always be based on the complete validated transaction *(Pending transaction + Validation stamp + Cross validation stamps)* thus allowing each node to check the validity of the transaction before storing it.

Technically only this validated transaction will be forwarded to the different storage nodes. In accordance with the *Principle 6* the storage nodes will perform the following tasks:

- The nodes of the "t*" storage pool of the "t" transaction will be responsible for retrieving and storing the other transactions of the chain and for exhaustive verification of the validity of the transaction and its associated chain. After verification, the nodes in the "t*" storage pool will store the complete chain in their local databases.
- The nodes of the storage pools associated with unspent outputs and recipients will store the transaction directly after verifying its unitary validity.
- The nodes of the associated beacon chain *(Formula 1.4.1)* will store the previous address of the chain, the transaction address and the timestamp to integrate them into the associated beacon chain.

1.3.3 Geographical coordinates vs. Network coordinates

The Uniris P2P network is based on two types of coordinates:

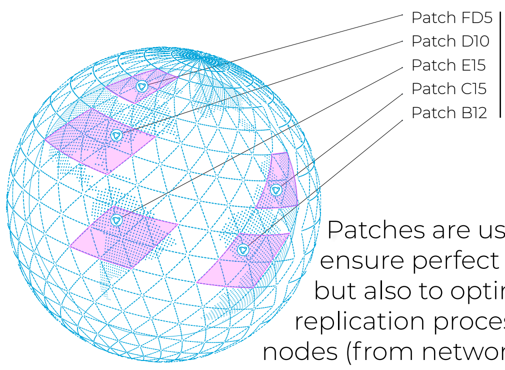
- Geographical Coordinates: Partially derived from the public IP address of the node.
- Network Coordinates: Calculated from modified algorithms similar to those published in *Vivaldi: A Decentralized Network Coordinate System* [1] and directly integrated in the beacon chain *(Beacons Chains – Figure 1.16)*.

To ensure the high availability and consistency of the data, the knowledge of the exact location of a node therefore becomes an essential factor, particularly to avoid data loss in the event of natural disasters. To achieve this, the network breaks down the coordinates of the nodes into two distinct parts:

- Network Coordinates of a node: these are computed globally and daily through *Beacon Chains (Figure 1.16)* and allows, cycle after cycle, to improve the accuracy of the network coordinates of the nodes. The position of a node is calculated from its latency (the minimum time to respond to a transaction being unmodifiable) and its throughput. This view is used to choose the best replication path *(Figures 1.10 and 1.11)*.
- Contextualized Geographic Coordinates of a node: the contextualization of the geographical coordinates of a node is realized at the time of update of its chain and by the nodes in charge of the validation of this update. This contextualisation is carried out on the basis of the deduced (GeoIP) or announced coordinates and is weighted by the calculations of the *Beacons Chains*. The update of these data being linked to the renewal of the keys of the nodes, they will thus be renewed automatically weekly and at each change of IP address.

The purpose of these additional checks is not to fault the nodes, because there are no associated punishments, but to ensure the best distribution in terms of geography and replication availability (1.3.2) from qualified data to ensure maximum availability of the data.

As shown in the figure below, these network and geographical coordinates are grouped by zones *(patches)* allowing a simplified calculation for nodes (number of zones, replication trees...). The zones are defined in a multiplet of 12 bits representing a tree (e.g. A5F).



To improve confidence in the geographical position of nodes, geolocation is verified by cross-checking using an adapted version of Vivaldi's algorithms

Patches are used inside mining and storage heuristics algorithms to ensure perfect geographical replication (data availability/security) but also to optimize network requests. During the validation/ replication process, the download will be performed from the nearest nodes (from network point of view), thus avoiding transfers that will cross the planet even though the data is already on a nearby node.

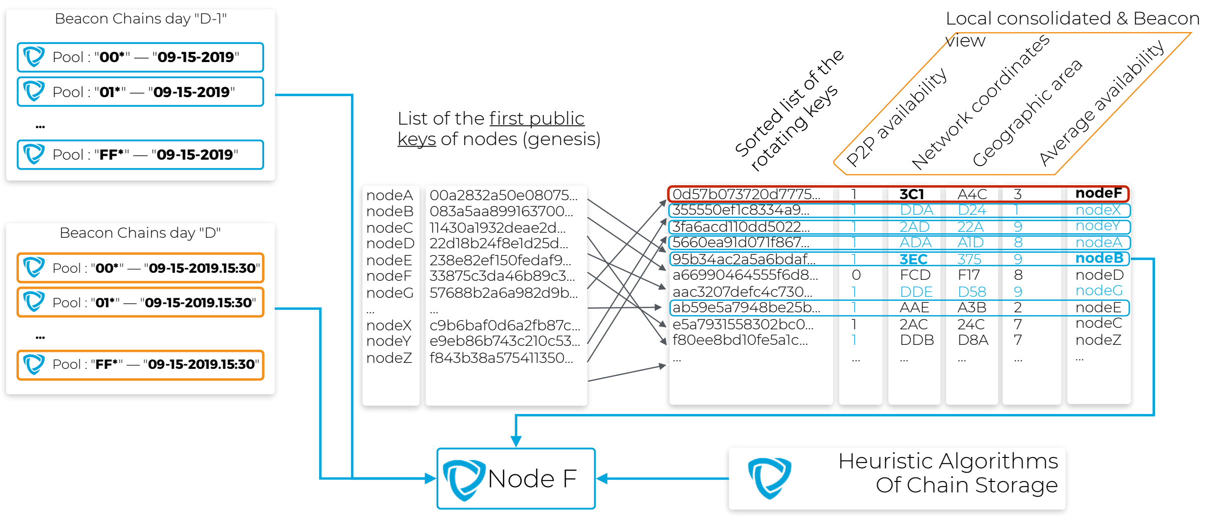
Figure 1.12: Geographical coordinates vs. Network coordinates

1.3.4 Self Repair and Reconfiguration of the network

To preserve an unlimited network capacity, a very small amount of information can be replicated on all nodes, and only the one that requires low writing (size or frequency). To ensure automatic repair and reconfiguration of replication trees, nodes use the same information and formulas as those used in the storage node election *(Figure 1.10)*. This mechanism allows any node to calculate in a few milliseconds the replication tree associated with an address and therefore to know if it should synchronize a chain or not *(Principle 6)*.

This process is executed autonomously by each of the nodes at 3 specific times:

- After a modification on Heuristic Algorithms (election and constraints on storage nodes, ex. update of the prediction module – Figure 1.21)
- After a change in the network nodes (disconnection or arrival of a new node on the network – Figure 1.14)
- Daily, after re-synchronization of local data with the latest Beacon Chains *(Figure 1.15)* by rebuilding the history of transactions chains (new transaction on a chain...)


Beacon Chains day "D-1"
Pool : "**00***" — "**09-15-2019**"
Pool : "**01***" — "**09-15-2019**"
...
Pool : "**FF***" — "**09-15-2019**"
Beacon Chains day "D"
Pool : "**00***" — "**09-15-2019.15:30**"
Pool : "**01***" — "**09-15-2019.15:30**"
...
Pool : "**FF***" — "**09-15-2019.15:30**"
List of the first public keys of nodes (genesis)
Local consolidated & Beacon view
Node F
Heuristic Algorithms Of Chain Storage

nodeA	00a2832a50e08075...
nodeB	083a5aa899163700...
nodeC	11430a1932deae2d...
nodeD	22d18b24f8e1d25d...
nodeE	238e82ef150fedaf9...
nodeF	33875c3da46b89c3...
nodeG	57688b2a6a982d9b...
...	...
nodeX	c9b6baf0d6a2fb87c...
nodeY	e9eb86b743c210c53...
nodeZ	f843b38a575411350...

Sorted list of the rotating keys	P2P availability	Network coordinates	Geographic area	Average availability	
0d57b073720d7775...	1	**3C1**	A4C	3	**nodeF**
355550ef1c8334a9...	1	DDA	D24	1	nodeX
3fa6acd110dd5022...	1	2AD	22A	9	nodeY
5660ea91d071f867...	1	ADA	A1D	8	nodeA
95b34ac2a5a6bdaf...	1	**3EC**	375	9	**nodeB**
a66990464555f6d8...	0	FCD	F17	8	nodeD
aac3207defc4c730...	1	DDE	D58	9	nodeG
ab59e5a7948be25b...	1	AAE	A3B	2	nodeE
e5a7931558302bc0...	1	2AC	24C	7	nodeC
f80ee8bd10fe5a1c...	1	DDB	D8A	7	nodeZ
...	...	...	...	...	...

Figure 1.13: Self Repair Mechanism of the Network

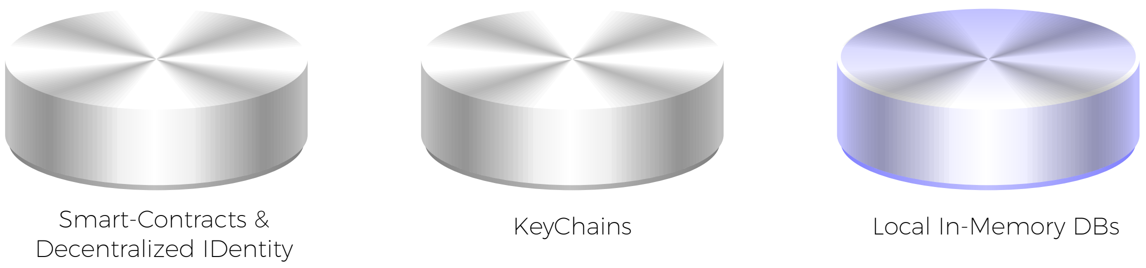
In the example in figure 1.13 above, the NodeF has been disconnected for 1 day and starts the bootstrapping process by downloading the beacon chains of the day "d-1" and the list of chained transactions of the day from the associated storage pools *(Figure 1.15)*. Once the context is downloaded, the NodeF will reconstruct its global and historical view of transaction chains (transactions, nodes, algorithm updates, etc.). Based on this view and all transactions that have not already been downloaded in the previous days, the NodeF will execute the Storage Heuristic Algorithms to know its position as a replica against the requirements of each transaction. Once this list is rebuilt, the NodeF will start downloading the missing transactions based on the network view using *Beacons Chains*, the NodeF will choose the NodeB to download the transaction, the network coordinates of the NodeB being the closest to its own.

1.3.5 Data Structure and Decentralized Ledgers

To provide the best balance between data availability, overall integrity of transaction chains and ledgers, each data type has a specific replication depth within the network. Since data that is often updated cannot be synchronized everywhere without generating network bottlenecks, very limited information is replicated on all nodes, such as *Network Transaction Chains – Principle 4* :

- Transactions chains of the authorized nodes (used for validation and replication node elections, but also to know their IP addresses)
- Transactions chains of Heuristic Algorithms and software *(Figure 1.5)*
- Shared Secrets Chains of Nodes and Devices *(Figure 1.19)*
- Prediction module Chain *(Figure 1.21)*.

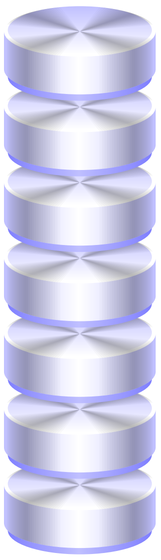
The replication depth strategy is defined by Heuristic Algorithms and executed directly by the decentralized software. This means that the network is autonomous to manage replication without using other external mechanisms. Using these algorithms, each node will also be able to autonomously re-synchronize the missing data for which it is in charge (new or disconnected nodes will change the position of the other nodes in the replication order – Figure 1.13).



Two decentralized databases constitute the references of the Uniris network:

- The first one: *Public Smart-Contracts and Decentralized Identity Ledgers* contains all user transaction chains (smart-contracts, ledgers movements & identities, etc.). This database is the heart of the solution, because it contains the Heuristic Algorithm chain *(Pinciple 5)* and all Network Chains (Authorized node chains, shared keys...) and consequently the governance rules.
- The second one, the *Keychains* manages decentralized wallets containing the private keys of users, groups, organizations and connected objects – *IoT*. The data is managed using algorithms similar to the storage of transaction chains. To secure the content of the wallets, the access is restricted to proof of possession of the private key (using signature) – the functioning of the *KeyChains* is described in *(Season 4)*.

Finally, nodes use local databases in memory, these databases are designed to be rebuilt each time a node is started and optimize the operations to be performed by each node (network view, node list, expenses/constraint checks, triggers, etc.). The figure below shows the main local databases in memory:



UCO Ledger	Optimized database focused on hosted UCO spent/unspent and conditions
NFT Ledger	Optimized database focused on Non-Financial Transactions (Emitter, spent/unspent & conditions)
MarketPlace Ledger	Optimized database focused on the stock management regarding MarketPlace hosted transactions
Triggers Ledger	Optimized database focused on smart-contract triggers for each hosted transaction
Network Ledger	Optimized database focused on Network requirements (Nodes, Sharedkeys, Prediction, etc.)
Pending&KO Ledger	Database listing all pending & KO transactions (to wait smart-contract countersignature ...)
P2P Ledger	Network discovery database (IP, availability, rating, geolocation, network coordinates,etc.)

In order to ensure the best possible organization of the data, and also to use as less memory as possible, these ledgers use a "column oriented " NoSQL database engine particularly adapted to field filtering, used in particular for calculations associated with Heuristic Algorithms.

1.4 Open, Optimized & Structured Distributed Network

There are two methods of communication within the distributed networks: the Gossip mode, whose properties are defined by the *knowledge of the outgoing neighbors*, which means each node of the network will discover the properties of the other nodes by interrogating them one by one, usually randomly, and the Broadcast mode [2] whose properties are defined by the *knowledge of the incoming neighbors*, which uses incoming connections. The Uniris network is a hybrid network that uses *Supervised Multicast* which is closer to the properties of Broadcast networks and combines the following properties:

Supervised Multicast occurs in three network processes:

- Transaction Replication Process: Capitalizing on incoming and outgoing connection information during the replication process *(Figure 1.17)*.
- During the process of updating *Network Transaction Chains* (Principle 6), for example, when updating the IP address of a node, the information is propagated to the entire network through the update of the chain associated with the node.
- By the decentralized process of *Beacons Chains – Figure 1.16* which, every 10min will take a snapshot and every day a synthesis of the state of each node to maintain a permanently global and qualified vision of the network.

Structured and Authenticated: Each node knows at any time the list of nodes allowed to participate in the network via the transaction chains associated with the nodes. Each connection is authenticated by the last public key of each node *(Principle 8)*.

Remunerated: Each node is remunerated according to its contribution to the network *(Incentive System)*, both for the validation phases and for the information provision phases: a node is not remunerated to replicate a transaction, but it will be when it makes the transaction available to the network.

Predictive and Adaptive: To compensate for the unpredictability of nodes on a network of ordinary nodes, the network has a prediction mechanism *(Figure 1.21)* that allows it to learn and provide countermeasures on anomalies detected.

Permissionless: any node can participate in the network as long as it has a *Hardware Security Module - HSM*[1] and they do not include any element related to a previous ban. The right to be a replica is open to all, but the right to validate (mining) is subject to public requirements *(Heuristic Algorithms)* based for example on geographical location or on taking into account the profitability of existing nodes to do not reduce the interest to participate in the network *(Season 3)*.

The schema below 1.14 represents the replication, bootstrapping and self-discovery mechanisms of the network:

[1]Cryptoprocessor used for the generation, secure sequestration of cryptographic keys and the calculation without disclosure of cryptographic algorithms (1.5.2)

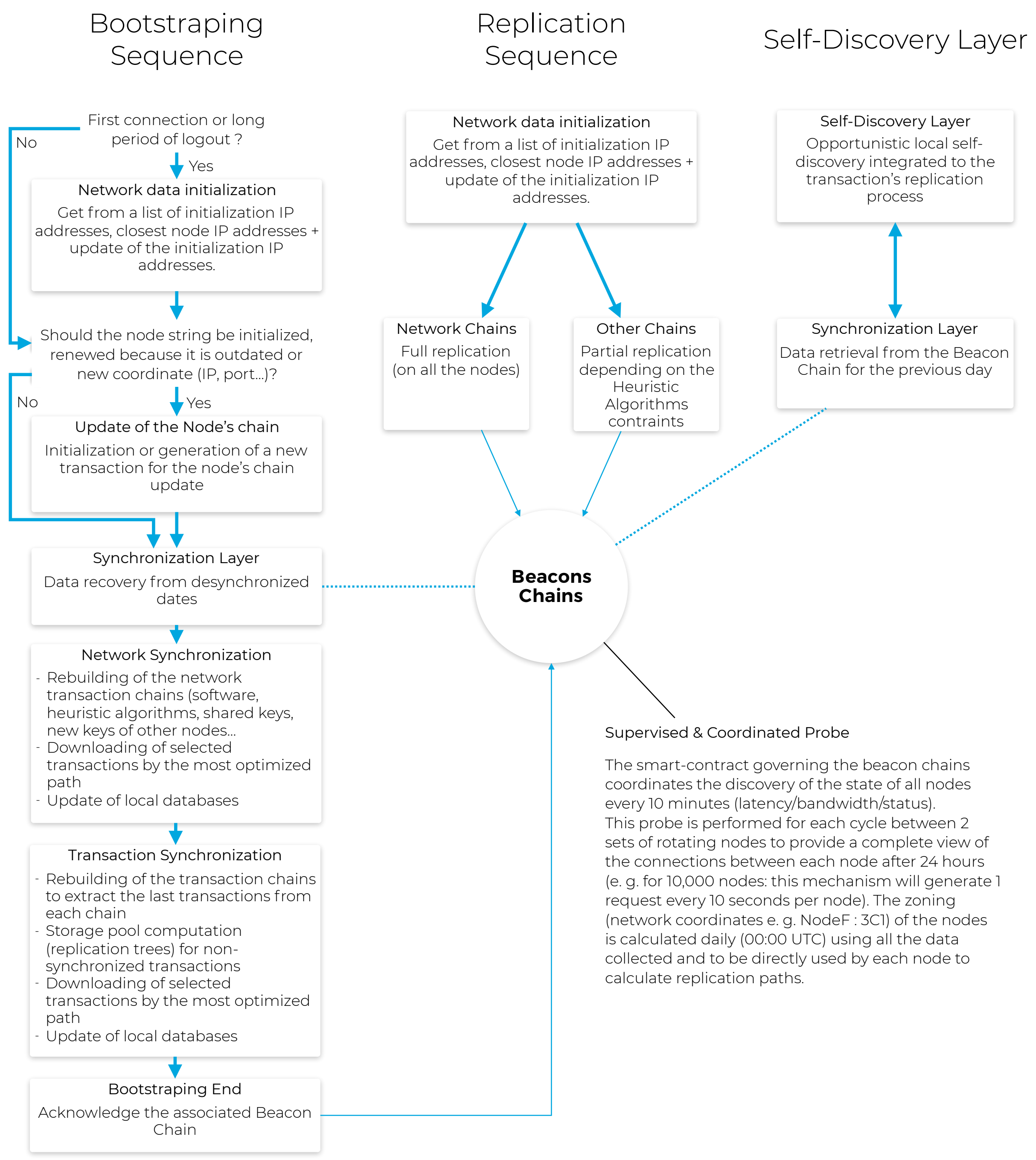

Bootstraping Sequence
Replication Sequence
Self-Discovery Layer
First connection or long period of logout ?
No
Yes
Network data initialization
Get from a list of initialization IP addresses, closest node IP addresses + update of the initialization IP addresses.
Should the node string be initialized, renewed because it is outdated or new coordinate (IP, port...)?
No
Yes
Update of the Node's chain
Initialization or generation of a new transaction for the node's chain update
Synchronization Layer
Data recovery from desynchronized dates
Network Synchronization
- Rebuilding of the network transaction chains (software, heuristic algorithms, shared keys, new keys of other nodes...
- Downloading of selected transactions by the most optimized path
- Update of local databases
Transaction Synchronization
- Rebuilding of the transaction chains to extract the last transactions from each chain
- Storage pool computation (replication trees) for non-synchronized transactions
- Downloading of selected transactions by the most optimized path
- Update of local databases
Bootstraping End
Acknowledge the associated Beacon Chain
Network data initialization
Get from a list of initialization IP addresses, closest node IP addresses + update of the initialization IP addresses.
Network Chains
Full replication (on all the nodes)
Other Chains
Partial replication depending on the Heuristic Algorithms contraints
Beacons Chains
Self-Discovery Layer
Opportunistic local self-discovery integrated to the transaction's replication process
Synchronization Layer
Data retrieval from the Beacon Chain for the previous day
Supervised & Coordinated Probe
The smart-contract governing the beacon chains coordinates the discovery of the state of all nodes every 10 minutes (latency/bandwidth/status).
This probe is performed for each cycle between 2 sets of rotating nodes to provide a complete view of the connections between each node after 24 hours (e. g. for 10,000 nodes: this mechanism will generate 1 request every 10 seconds per node). The zoning (network coordinates e. g. NodeF : 3C1) of the nodes is calculated daily (00:00 UTC) using all the data collected and to be directly used by each node to calculate replication paths.

Figure 1.14: Bootstrapping, Replication & Self-discovery

In the following sections, we will elaborate these concepts:

1.4.1 Beacon Chains Responsible for Global Synchronization

Computing Beacon Chains addresses Since no node has the physical ability to know the status of each transaction in an unlimited network, the Uniris network uses a set of specific transaction chains each containing a subset of the addresses of the last transactions (00*, 01*... FF*) for a given date. The transaction address is determined directly from the date and address subset (via a derivation function and a seed) in a similar way to the following calculation:

$$\text{PrivateKey}_{(\text{subset + date})} = \text{Hash}(\text{subset + date, Storage Nonce})$$
$$\text{Address}_{(\text{subset + date})} = \text{Hash}(\text{PubKey}(\text{PrivateKey}_{(\text{subset + date})}))$$

Example: Full day for 00 subset on 2019, April 18th

$$\text{Address}_{(00 + 2019.04.18)} = \text{Hash}(\text{PubKey}(\text{PrivateKey}_{(00 + 2019.04.18)}))$$

Example: Current day (every 10min) for 00 subset on 2019, April 18th at 2:40

$$\text{Address}_{(00 + 2019.04.18\ \ 2:40)} = \text{Hash}(\text{PubKey}(\text{PrivateKey}_{(00 + 2019.04.18\ 2:40)}))$$

This derivation key is therefore known by all nodes as soon as they join the network. This mechanism thus allows nodes to find at any time the transaction chain or directly the transaction containing all the transactions of a given date.

Transaction Tracking and Time-Stamping The transmission of the validated transaction address to the associated storage subset (figures 1.10 et 1.11) is performed jointly between the validation nodes. To insure confidentiality of the transactions within a chain, the address of the previous transaction is encrypted with the nodes shared key. Thus allowing nodes to update their replication table by locally reconstructing the transaction history *(Figure 1.13)*, without disclosing transactions associated with a chain.

The figure 1.15 represents the mechanism of generating the chains of the subsets every 10 minutes as well as the generation of the last transaction which will be the consolidation of all the information collected during the day for a given subset. Even if all the chains are cross-linked by the signing of previous transactions, each day will have its own transaction chain and therefore different storage pools, thus making it possible to balance the storage on each node. The list of transactions is stored in the zone $\boxed{\text{DATA/Content}}$.

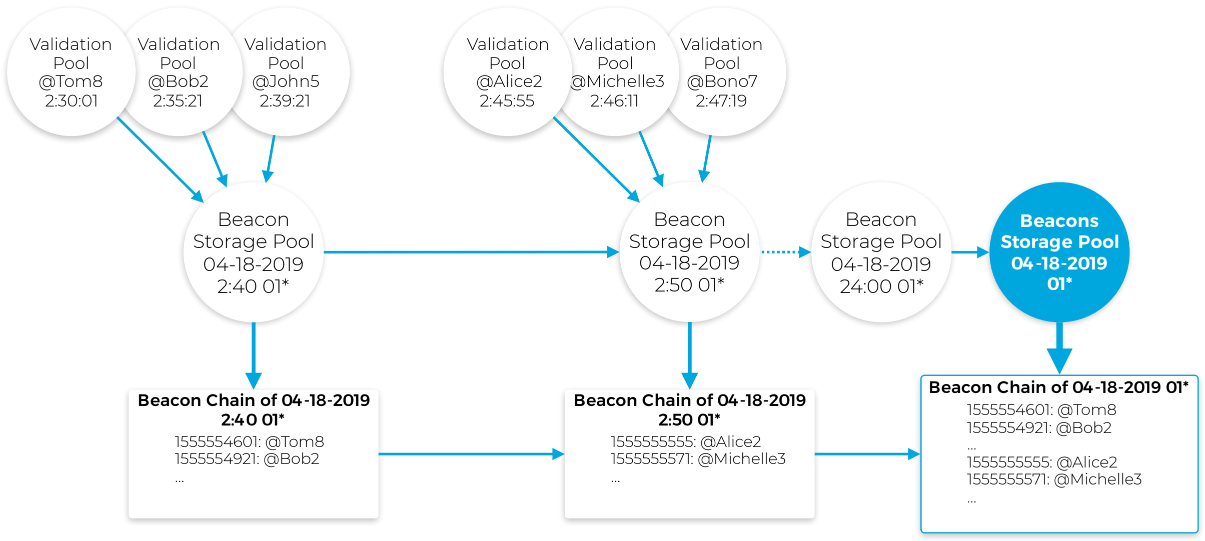
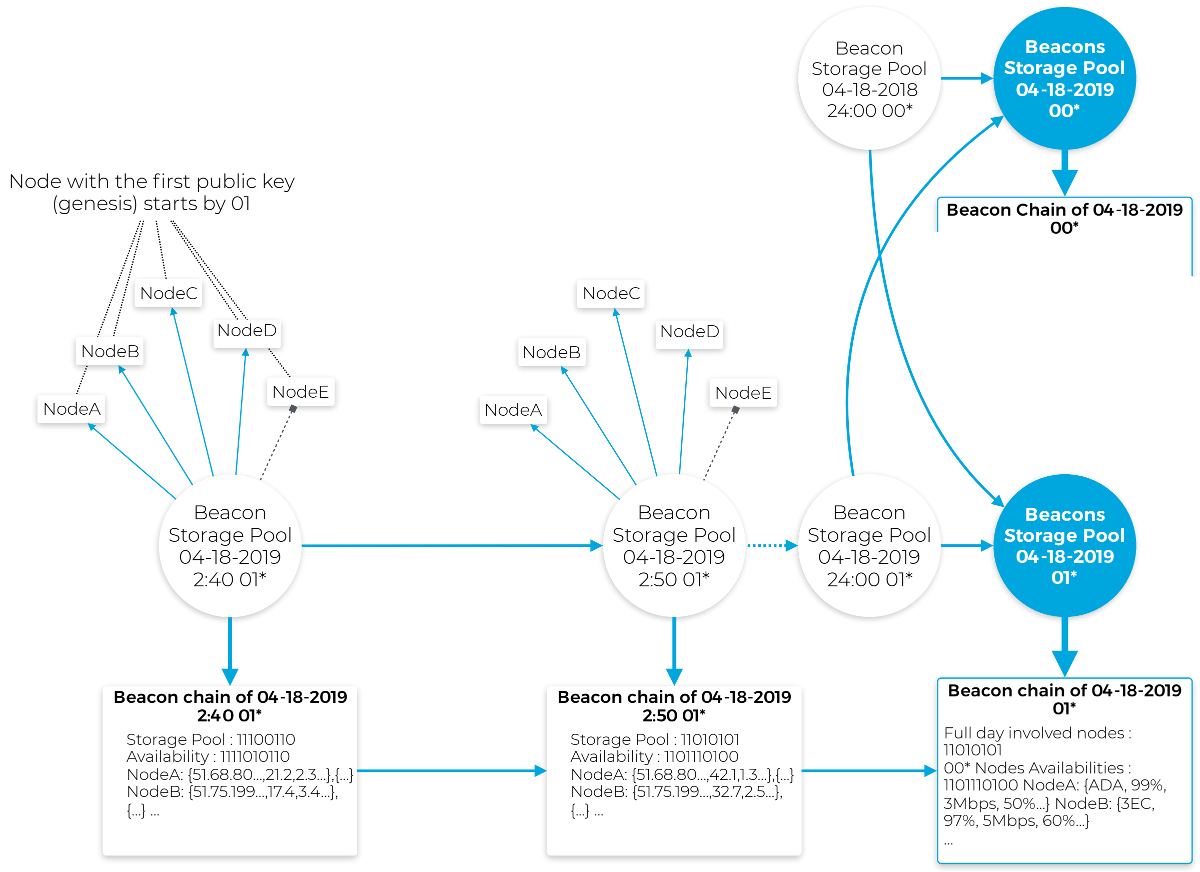

Validation Pool @Tom8 2:30:01
Validation Pool @Bob2 2:35:21
Validation Pool @John5 2:39:21
Validation Pool @Alice2 2:45:55
Validation Pool @Michelle3 2:46:11
Validation Pool @Bono7 2:47:19
Beacon Storage Pool 04-18-2019 2:40 01*
Beacon Storage Pool 04-18-2019 2:50 01*
Beacon Storage Pool 04-18-2019 24:00 01*
Beacons Storage Pool 04-18-2019 01*
Beacon Chain of 04-18-2019 2:40 01*
1555554601: @Tom8
1555554921: @Bob2
...
Beacon Chain of 04-18-2019 2:50 01*
1555555555: @Alice2
1555555571: @Michelle3
...
Beacon Chain of 04-18-2019 01*
1555554601: @Tom8
1555554921: @Bob2
...
1555555555: @Alice2
1555555571: @Michelle3
...

Figure 1.15: Beacon Chains Transaction Tracing and Time-Stamping

Status and Network Coordinates of nodes Beacon chains also contain the network status for the subset of nodes whose first public key (genesis) belongs to the same subset (01*). For each new Beacon transaction on this chain (ie. "01*" on "2019-04-18" at "00:20") the associated storage nodes (Beacon Pool) will have the task of:

- Checking the status of the nodes belonging to this subset and store the result obtained by consensus in binary form (the list of nodes being known: e.g. 111010101 for the status of the first 9 nodes in the ordered list).
- Listing the nodes that participated in this verification (always in binary form from the sorted list).
- And among other information, latency and throughput between each of the nodes in the storage pool and each of the nodes in the subset (01*). This information is used to determine the network coordinates (1.3.3) of each node at the end of the day in a synthesis transaction that will be downloaded by all the nodes. During this computation, as shown in the figure 1.16, each of the "end-of-day" storage pools will retrieve all the beacon chains from each of the subsets to globally calculate the network positioning of each of the nodes and integrate this list (complete and locally calculated) into the synthesis transaction.


Node with the first public key (genesis) starts by 01
NodeA
NodeB
NodeC
NodeD
NodeE
Beacon Storage Pool 04-18-2019 2:40 01*
Beacon Storage Pool 04-18-2019 2:50 01*
Beacon Storage Pool 04-18-2019 24:00 01*
Beacons Storage Pool 04-18-2019 01*
Beacon Storage Pool 04-18-2018 24:00 00*
Beacons Storage Pool 04-18-2019 00*
Beacon Chain of 04-18-2019 00*

Beacon chain of 04-18-2019 2:40 01*
Storage Pool : 11100110
Availability : 1111010110
NodeA: {51.68.80...,21.2,2.3...},{...}
NodeB: {51.75.199...,17.4,3.4...},
{...} ...

Beacon chain of 04-18-2019 2:50 01*
Storage Pool : 11010101
Availability : 1101110100
NodeA: {51.68.80...,42.1,1.3...},{...}
NodeB: {51.75.199...,32.7,2.5...},
{...} ...

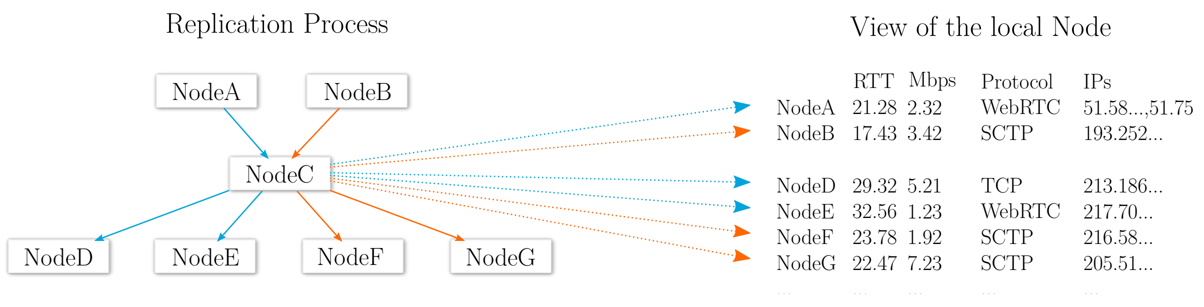
Beacon chain of 04-18-2019 01*
Full day involved nodes : 11010101
00* Nodes Availabilities : 1101110100 NodeA: {ADA, 99%, 3Mbps, 50%...} NodeB: {3EC, 97%, 5Mbps, 60%...}
...

Figure 1.16: Beacon Chains Status and Network Coordinates of Nodes

Unlimited Network Paradigm: The segmentation of the subsets will be adjusted by the Heuristic Algorithms according to the number of transactions per second supported by the system. For example, a subset created from the first byte ("00*") will generate a request every 2.56 seconds on each of the pool nodes for a global load of 100 req/sec on the network, similarly a subset created from the first three bytes ("000000*") will generate a request every 5.59 seconds on each of the pool nodes for a global load of 3,000,000 req/sec.

1.4.2 Network Self-Discovery Layer

As described in the figures 1.14 and 1.17, the "semi-passive" or "opportunistic" self-discovery operation will allow each node to build a local view on the state of the nearby nodes without generating any new transactions. Network data (IPs, protocols, latency, throughput, etc.) can be calculated directly and other data (disk usage, average CPU/memory usage) will be explicitly transmitted by the transaction's transmitting nodes. This local view (see figure 1.17) will then be compared with the views of the other nodes when updating the beacon chains *(Figure 1.16)*.



	RTT	Mbps	Protocol	IPs
NodeA	21.28	2.32	WebRTC	51.58...,51.75
NodeB	17.43	3.42	SCTP	193.252...
NodeD	29.32	5.21	TCP	213.186...
NodeE	32.56	1.23	WebRTC	217.70...
NodeF	23.78	1.92	SCTP	216.58...
NodeG	22.47	7.23	SCTP	205.51...
...	...	...	...	...

Figure 1.17: Discovery during the Replication Phase

1.4.3 Network Bootstrapping Layer

When a node or client first connects to the network, the node or client will need to contact one of the pre-filled "Seed" (list of stable IP addresses of nodes with higher availability) which will return the updated list of "Seeds" and the list of the nearest available nodes (via *Geographical coordinates vs. Network coordinates*).
In the case of the first connection of a node, it will generate a transaction to initialize its chain toward one of the nearest nodes so that it can then be validated and completed with the necessary information to participate in the network (Node Shared Secrets...).
A node that has already been initialized, but whose chain has expired or one of the network information has been modified (IP, port, protocol, etc.) must also start the sequence by updating its chain.

Once the node chain is initialized or updated, the node will be able to request the list of network nodes from any other node. This list contains for each node of the network: its identifier, its first (genesis) and last public key, IP address, port, protocol, best rate, best latency, geographical area, network coordinates and its system data (disk usage and average CPU/RAM availability). This view is built by each of the nodes from the qualified data of the previous day's beacon chains and is kept up to date through the process of network chains replication (all of which is replicated throughout the network).

Once the minimum information has been restored, the node can then re-synchronize all the data as follows:

1. Calculation of beacon chain addresses *(Formula 1.4.1)* from the dates when the information was not synchronized and retrieval of the respective associated transactions.
2. Rebuild the history of the network transactions chains (node chains, heuristic algorithms, etc.), retrieve associated chains and update local databases.
3. Reshape the history of other transaction chains, calculating replication trees, identifying transactions to download (Figure 1.13) and retrieving transactions by the most optimized paths (via network coordinates 1.3.3).
4. Finally, when all the data has been updated, the node will notify the beacon chain associated with its "genesis" public key so that it can once again participate in the network.

Note: *In order to facilitate the re-synchronization operation by the nodes, the validation pool in charge of updating the node key will specify the date from which the node will be able to participate in the network. To define the required duration for a node to re-synchronize its data, the prediction module is learning depending on the desynchronization status, hardware and network capacities of the node and from the previous beacon chains notification to precise the estimation.*

1.5 Security Beyond Known Solutions

The Uniris network has been designed to provide a significant improvement in IT security for users (biometric identification without any key storage *(Season 5)*), and also to significantly improve security on decentralized networks without affecting the fundamentals.

The following sections will define:

> 1.5.1 Network Resilience & Malicious Operations Detection
> 1.5.2 Adaptive and Quantum-Safe Security
> 1.5.3 Separation of Powers to Strengthen Security

Despite the obvious advantages of distributed networks over centralized systems that are based entirely on the trust of third parties, it is still important to list the known weaknesses of these networks:

51% Proof-of-Work flaw: For proofs of work based on computing power (HashRate), the flaw that can be exploited by an attacker, is to gather during a given period a computing power equivalent to the power of the entire network. The security of the network is at mercy of the computing power of its nodes.

Proof-of-Stake flaws: The main risks for Proof-of-Stake, beyond the somewhat recentralized nature of consensus, are taking control by financial means and for an attacker to be able to concentrate his attack on a few nodes of the network to take control.

Faults of the dBFT The networks using Delegated Byzantine Fault Tolerance consensus are not fully decentralized. Moreover dBFT provides fault tolerance of, f=(n-1)/3 dishonest nodes where n is the total number of nodes. The idea of consensus being ruled by $2/3^{rd}$ majority.

Split-Brain flaws: The split-brain can occur in a non-fraudulent way, for example after a submarine cable is cut, making one half of the network inaccessible to the other half. In this case, two versions of chains could exist during the break.

DDoS vulnerabilities: The DDoS vulnerability on a decentralized network, consists of overloading all or part of the nodes with transactions, thus incapacitating a part of the network to participate in consensus building.

Sybil attack flaws: The Sybil attack (personality duplication) aims to overload the network with false information from multiple identities or from of a multiplied identity, this attack is even more effective on unstructured networks using p2p information from neighboring nodes to create their views.

Smart-Contract (VM) faults: The most well known is "TheDAO" fault that appeared in May 2016 and which resulted in the creation of two separate block chains on the Ethereum network. This vulnerability highlighted the problem of code being executed blindly by virtual machines on nodes. As such, other Blockchains, for example, propose a method of mandatory programming of smart-contracts to improve determinism on the output result, but consensus is generally based on Proof-of-Stake or dBFT.

Wallet Flaws: Wallets (wallets managers) can be summarized into three main families:

Outsourced Wallets: Generally provided by trading platforms, they pose the problem of key centralization and are therefore prime targets for an attacker (Zaif, Coincheck), zero-day attacks being much easier than taking control of tens of thousands of minors.

Wallet Apps: Provide real comfort for the users who can use their cryptographic keys anywhere, but these apps expose problems of security of the system layer, the security of other installed applications, and also the way they are encoded (e.g. Bitcoin-Qt which raised the problem of the predictability of generated keys).

Hardware Wallets: Physical wallets provide the best way to secure cryptographic keys, but they also pose the widespread problem of media loss.

The vulnerabilities listed cover four areas, on which the Uniris network provides the following countermeasures:

Consensus Layer where the best consensus offers resistance to attacks of the order of 66% of malicious or dishonest nodes, thanks to the ARCH consensus Uniris network can reach a risk of 10^{-9} (1/1,000,000,000) below aeronautical or nuclear standards even with a network composed of more than 90% of malicious nodes *(Figure 1.18)*. The ARCH Consensus is based on Atomic Validation or the formal consensus obtained by Rotating Heuristic Election of validation nodes – meaning total agreement on the validation of a transaction by the nodes selected by a random and rotating election.

Network Layer where the majority of Blockchain solutions are based on Gossip, the Uniris network is based on Supervised Multicast *(Figure 1.14)* allowing to avoid unnecessary network traffic while providing a complete and shared view of the state of the nodes. This capacity allows the network to detect any Split-Brain type problem and provide the appropriate countermeasures. All transmissions being authenticated at the client level (shared keys of sending devices and software) or at the node level (node chains and Shared Secrets) any DDoS attack could quickly be identified and ruled out.

Smart-Contracts Layer the Bitcoin network relies on unspent transaction outputs (UTXOs) allowing everyone to check the consistency between the different inputs on the ledger and the outputs. Since each transaction is signed cryptographically, it is impossible to create a fake transaction on a Wallet whose private key is unknown. The Uniris network is based on the same property: only validated transactions are authoritative, it is not possible, for example, to make a database inside a smartcontract. On the other hand, all current smart-contract use-cases would still work as well on the Uniris network as on other networks *(for example, in an e-commerce smartcontract, the smart-contract issued by a merchant will be able to define stocks, prices and interactions with its customers using a view which is continuously updated by the nodes responsible for storing the smart-contract and based on transactions validated to that same smart-contract – (Season 2))*. The "UTXO" operation does not give a status within a smart-contract but allows it to be calculated (in the example above the merchant cannot directly query a smart-contract on the status of orders, but can verify the proven status of orders through validated transactions).

Wallet Layer although discussed lastly, the primary goal of the Uniris network was to develop a biometric identification method allowing users to retrieve their cryptographic keys using the body and without ever having to store these biometric or derived data, outside the body *(Season 5)*. When combined with the Uniris network, the mechanism of device level authorization is such that an attacker will not be able to generate a transaction even if the user private keys were somehow compromised. The objective is to remove the adoption barriers associated with managing cryptographic keys and securing them for any human being.

1.5.1 Network Resilience & Malicious Operations Detection

The properties of the ARCH requiring 100% positive and matching responses, the network allowing the authentication of each of the nodes and the supervised multicasting ensuring reliable knowledge of the availability of the nodes - makes it possible for the consensus to satisfy the hypergeometric distribution law (1.5). The hypergeometric distribution describes the probability of k success (detection of fraudulent operation) for n drawn (verifications) without repetition with a total finite number of nodes N and by considering a number N1 of malicious nodes (90%):

$$\mathbb{P}[X = k] = \sum_{k=1}^{p} \frac{\binom{N_1}{k} \times \binom{N-N_1}{n-k}}{\binom{N}{n}} \tag{1.5}$$

This law thus makes it possible to estimate the chance of detecting a malicious operation by knowing the number of validations (verification). The example below calculates that for a total of 100 nodes, including 90 malicious ones, 42 checks give a 99,7% chance of detecting a fraudulent transaction (having at least one honest node $1 \leqslant k \leqslant 10$).

$$\mathbb{P}[1 \leqslant X \leqslant 10] = \sum_{k=1}^{10} \frac{\binom{10}{k} \times \binom{90}{42-k}}{\binom{100}{42}} \approx 99.7\% \tag{1.6}$$

In the same manner and on the same drawing, 84 verifications will allow to obtain 99,9999999% of chance to detect a fraudulent operation or the risk of not detecting is 10^{-9} *(these are beyond the standards of the acceptable risk for aviation or nuclear)* :

$$\mathbb{P}[1 \leqslant X \leqslant 10] = \sum_{k=1}^{10} \frac{\binom{10}{k} \times \binom{90}{84-k}}{\binom{100}{84}} \approx 99,9999999\% \tag{1.7}$$

The hypergeometric distribution becomes really interesting when the total number of nodes in the network increases (considering 90% of dishonest nodes in above equations). The figure 1.18 below shows the number of checks required (number of cycles) as a function of the number of nodes to obtain a risk in the order of 10^{-9}. Thus, for 100 nodes, it will require 84 validations (84%), for 1000 nodes: 178 (or 17.8%), for 10,000 nodes: 195 (or 1.9%) and for 100,000 nodes: 197 (or 0.2%)

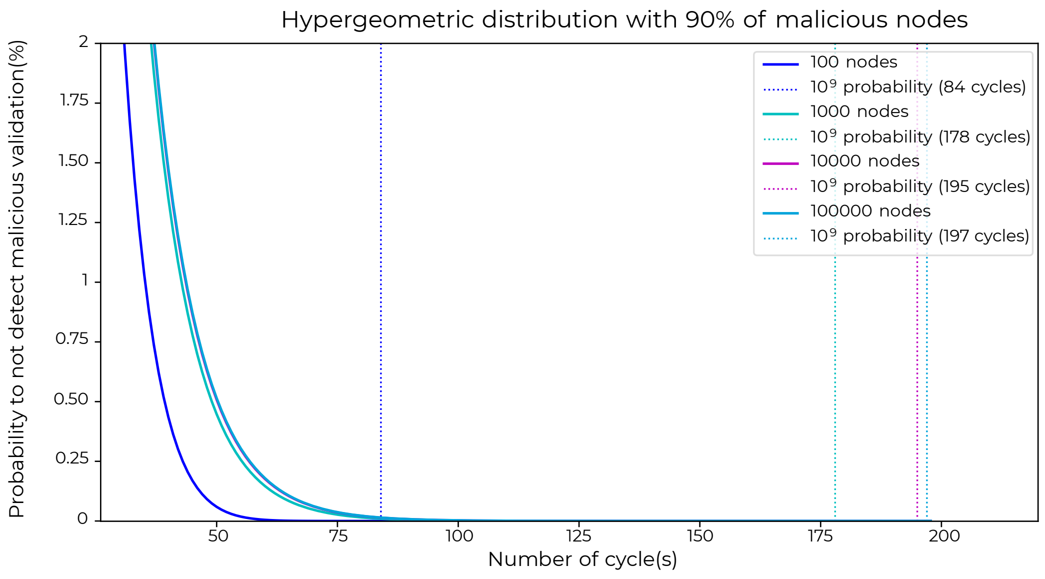


Figure 1.18: Hypergeometric distribution for 100, 1000, 10000 and 100000 nodes

$$1 - \left[\lim_{N \to +\infty} \mathbb{P}[X = k] \right] = 1 - \left[\lim_{N \to +\infty} \sum_{k=1}^{p} \frac{\binom{0.1*N}{k} \times \binom{0.9*N}{n-k}}{\binom{N}{n}} \right] \approx 10^{-9} \Rightarrow n \approx 200 \qquad (1.8)$$

This means that even with 90% of malicious nodes, no matter how many nodes are running on the network, a control with a tiny part of the network (less than 200 nodes) ensures the atomicity property of network transactions.

1.5.2 Adaptive and Quantum-Safe Security

To allow cryptographic backward compatibility, to evolve the network as the cryptographic research progresses and to provide the choice of cryptographic algorithms to people, organizations or countries, the public key will be versioned on a byte indicating the algorithm used. The supported algorithms are listed in a chain of specific smart-contracts.
Combined with the non-disclosure mechanism of public keys *(Remark 2)* and combined with the need to simultaneously known multiple temporary private keys (1.2) to be allowed to generate a transaction, the task of a quantum computer potentially capable of "breaking" private keys should be considerably more complex.
Except for hardware compatibility issues (HSM, etc.), EdDSA Signatures, Curve25519 and AES256 will be used by default on the network.

1.5.3 Separation of Powers to Strengthen Security

The Uniris network is based on the concept of SoD *(Segregation of Duties)*. To achieve a formal separation of powers between miners themselves and between users and miners, the Uniris network uses the technical requirement of knowing a group of cryptographic keys (one group knows the private key while another only knows the public key and vice versa).

Knowledge of shared secrets makes it possible for nodes to participate in the validation process of a transaction (mining) and therefore be paid for this work. Technically, the shared secret is a derived or stretched key, generated from a key derivation function [3] [4] allowing nodes to retrieve the shared key from a secret (Seed), a derivation function and, in the case of shared node secrets, the date. The secret (Seed) and the derivation function are encrypted with the last known public key of each node and renewed each time a node is banned from the network (see figure below).
The renewal of the shared key of the nodes is performed daily at 00:00 UTC and is communicated to the rest of the network in the minutes before.

By using the mechanism of self-triggered smart-contracts *(Season 2)*, the daily generation of a new transaction on the Shared Secrets chain (Figure 1.19: @NodeShardKey-3) is entrusted to the first available storage node associated with this address (1.3.2) and is unpredictably validated by the elected validation nodes (1.2.2). To preserve the integrity of this chain even in the event of a private key disclosure, this one uses the inherited recursive constraints mechanism *(Season 2)* prohibiting any modification of the chain regardless of the private key's knowledge.

The figure 1.19 below represents the renewal of *Shared Secrets* after banning one or more nodes from the network:

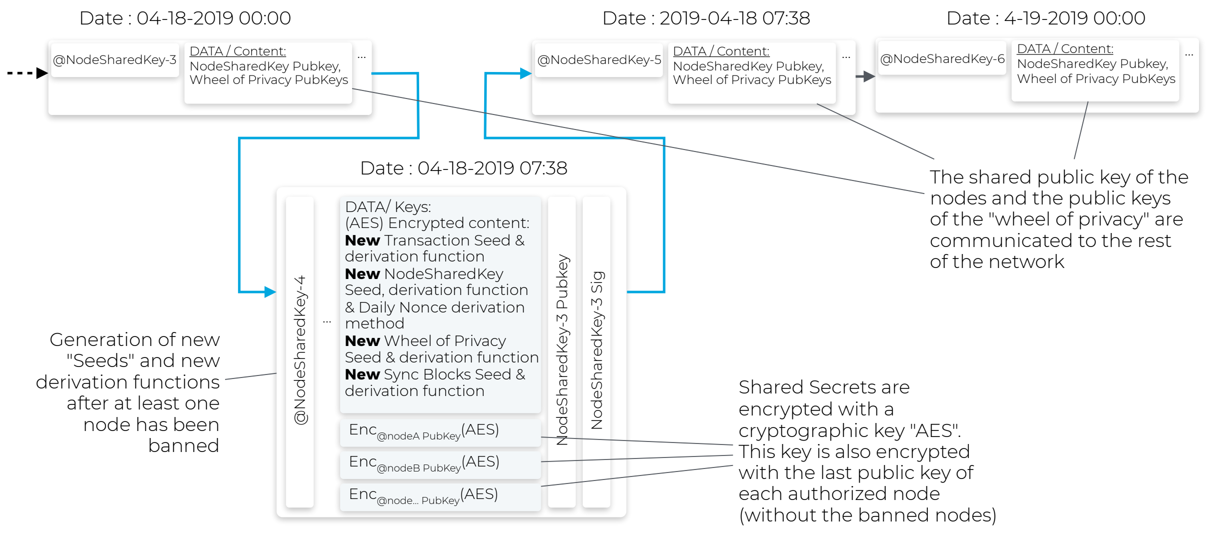

Date : 04-18-2019 00:00
@NodeSharedKey-3
DATA / Content: NodeSharedKey Pubkey, Wheel of Privacy PubKeys
Date : 2019-04-18 07:38
@NodeSharedKey-5
DATA / Content: NodeSharedKey Pubkey, Wheel of Privacy PubKeys
Date : 4-19-2019 00:00
@NodeSharedKey-6
DATA / Content: NodeSharedKey Pubkey, Wheel of Privacy PubKeys
Date : 04-18-2019 07:38
@NodeSharedKey-4
DATA/ Keys: (AES) Encrypted content: New Transaction Seed & derivation function New NodeSharedKey Seed, derivation function & Daily Nonce derivation method New Wheel of Privacy Seed & derivation function New Sync Blocks Seed & derivation function
Enc@nodeA PubKey(AES)
Enc@nodeB PubKey(AES)
Enc@node... PubKey(AES)
NodeSharedKey-3 Pubkey
NodeSharedKey-3 Sig
The shared public key of the nodes and the public keys of the "wheel of privacy" are communicated to the rest of the network
Generation of new "Seeds" and new derivation functions after at least one node has been banned
Shared Secrets are encrypted with a cryptographic key "AES". This key is also encrypted with the last public key of each authorized node (without the banned nodes)

Figure 1.19: Shared or Crossed Secrets of Nodes

1.6 A Network Capable of Reconfiguring Itself to Prevent Disasters

To enable a decentralized network to survive for decades or centuries, it must be able to adapt to threats and react accordingly. For this, the Uniris network is based on two adaptive algorithms:

Action and reaction capacity: This capacity is ensured by Heuristic Algorithms which manage the majority of network behavior: for example by adapting the constraints on the number of validations/replications according to the number of connected nodes of the network or in weighting a geographical area according to the average availability of the nodes. All behaviors are pre-established directly in the algorithms.

Prediction, anticipation and correction capacity This module has the ability to link patterns/samples and futures behaviour, hence potentially predict a future state of the network. For example, to detect that an operator in a given country updates the firmware of his boxes monthly and that the network will be cut for all associated nodes during this period, that the electricity network for a given region of the world is cut during storms or even before an attempt to attack the nodes in charge of the validation will tend to respond less quickly to other transactions. The objective is to guarantee at all times the availability of data, for example by changing the availability weight of one of the nodes elected for replication or by increasing the number of validations required for a given transaction without reducing the constraints of Heuristic Algorithms.

The following sections will define:

1.6.1 Uniris Oracles Chains

To reconstitute the context of events before the anomaly occurs, the network must have as much qualitative information as possible. For this purpose, the prediction module uses two decentralized mechanisms:

Beacons Chains: This lists all network states and transactions every 10 min and summarises it every day, its operation is described in the section *Beacon Chains Responsible for Global Synchronization*.

Oracle: The *"World State"* Oracle behaves in the same way as the *Beacons Chains* except that new transactions on the chain are not generated every 10 min, but every time information is updated (for example when a new weather report is published) - all information will also be aggregated at the end of the day (00:00 UTC) by a last transaction on the chain. The information listed within this chain is of several kinds: climatic, financial (stock market prices, crypto-currencies prices – including the Uniris Coin), societal (number of occurrences of keywords on information sites or even when possible the last words most used on search engines). All references (URLs, etc.) are listed in a specific smart-contracts chain.

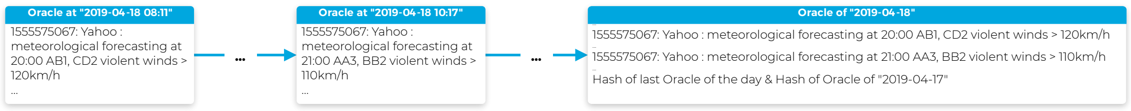

Oracle at "2019-04-18 08:11"
1555575067: Yahoo : meteorological forecasting at 20:00 AB1, CD2 violent winds > 120km/h
...
...
Oracle at "2019-04-18 10:17"
1555575067: Yahoo : meteorological forecasting at 21:00 AA3, BB2 violent winds > 110km/h
...
...
Oracle of "2019-04-18"
1555575067: Yahoo : meteorological forecasting at 20:00 AB1, CD2 violent winds > 120km/h
1555575067: Yahoo : meteorological forecasting at 21:00 AA3, BB2 violent winds > 110km/h
Hash of last Oracle of the day & Hash of Oracle of "2019-04-17"

Figure 1.20: Oracle's chain representation

The purpose of using these data is to reconstitute the context before the event period in such a way as to detect early signs.

1.6.2 Prediction Module's Parameters

The parameters of the prediction module are stored directly on the smart-contracts chain *Uniris Prediction Module*. This autonomous smart-contract will be able to permanently modify its chain, without having the capability to modify the constraints imposed *(Anomalies Triggers, Scope of countermeasures proposals & KPI)* which will be protected by inherited recursive constraints (22) imposing a community vote *(Season 2) (Season 3)*. The prediction module has 5 main parameters :

Anomalies Triggers: Unlike *Beacon Chains* and *Oracle Chains*, anomalies do not provide a learning context, only the events and thresholds to be monitored that trigger the pattern detection phase (scheme that seems to have led to the anomaly). This anomaly detection is performed locally by each node, especially during the network self-repair cycles (1.3.4). For instance, when a transaction will have a critically low level of replication after the unavailability of a large number of nodes on one or more geographical areas.

Scope of Countermeasures Proposals: The countermeasures perimeter allows nodes to know on which parameters countermeasures will be applied (e.g. request an additional geographical replication area for transactions whose storage nodes are located on a storm's trajectory).

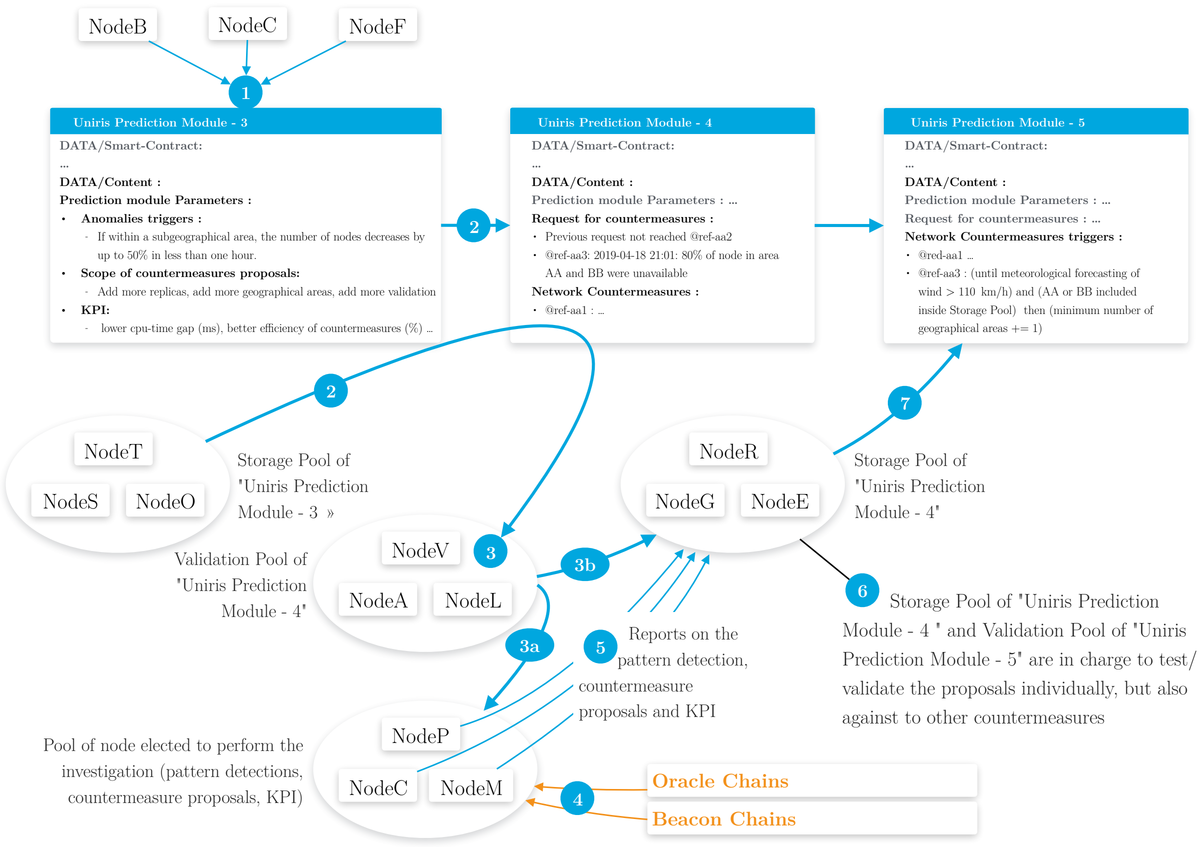
KPI: The key performance indicator makes it possible to weight each of the countermeasure proposals, in particular to ensure a minimum cost for the network, for example: the additional calculation time, the number of replicas, the number of additional validations induced, the anticipation in allowed time and of course the success rate of the proposed countermeasure.

Request for Countermeasures: They represent network requests related to qualified anomalies (having received a minimum number of nodes confirming the anomaly), anomalies without relevant countermeasure proposals (KPI efficiency factors) will remain listed in this section.

Network Countermeasures Triggers: These triggers are used continuously by Heuristic Algorithms especially for validation and storage node elections, but especially during the self-repair phase (1.3.4) thus allowing the network to apply countermeasures before an event even occurs.

1.6.3 Functioning of the Prediction Module

The prediction module is hosted on the smart-contracts chain *Uniris Prediction Module* replicated on all network nodes. Technically, the content of the triggers (anomaly and countermeasure triggers) is added to the local trigger database *Triggers Ledger* (1.3.5) and will be checked for each operation. The diagram below shows in a simplified way the mechanism for modifying the prediction model following the detection of a qualified anomaly.



NodeB
NodeC
NodeF
1
Uniris Prediction Module - 3
DATA/Smart-Contract:
...
DATA/Content :
Prediction module Parameters :
• Anomalies triggers :
- If within a subgeographical area, the number of nodes decreases by up to 50% in less than one hour.
• Scope of countermeasures proposals:
- Add more replicas, add more geographical areas, add more validation
• KPI:
- lower cpu-time gap (ms), better efficiency of countermeasures (%) ...
2
Uniris Prediction Module - 4
DATA/Smart-Contract:
...
DATA/Content :
Prediction module Parameters : ...
Request for countermeasures :
• Previous request not reached @ref-aa2
• @ref-aa3: 2019-04-18 21:01: 80% of node in area AA and BB were unavailable
Network Countermeasures :
• @ref-aa1 : ...
Uniris Prediction Module - 5
DATA/Smart-Contract:
...
DATA/Content :
Prediction module Parameters : ...
Request for countermeasures : ...
Network Countermeasures triggers :
• @red-aa1 ...
• @ref-aa3 : (until meteorological forecasting of wind > 110 km/h) and (AA or BB included inside Storage Pool) then (minimum number of geographical areas += 1)
2
7
NodeT
NodeS
NodeO
Storage Pool of 'Uniris Prediction Module - 3 »
NodeR
NodeG
NodeE
Storage Pool of 'Uniris Prediction Module - 4'
NodeV
3
NodeA
NodeL
Validation Pool of 'Uniris Prediction Module - 4'
3b
6
Storage Pool of 'Uniris Prediction Module - 4 ' and Validation Pool of 'Uniris Prediction Module - 5' are in charge to test/validate the proposals individually, but also against to other countermeasures
3a
5
Reports on the pattern detection, countermeasure proposals and KPI
Pool of node elected to perform the investigation (pattern detections, countermeasure proposals, KPI)
NodeP
NodeC
NodeM
4
Oracle Chains
Beacon Chains

Figure 1.21: Prediction Module

1 The anomaly is detected by several nodes that will transmit a new transaction to the smart-contracts chain *Uniris Prediction Module*
2 Once the quorum of the number of node confirmations has been reached, the storage pool in charge of storing the smart-contract *Uniris Prediction Module - 3* will generate an update of the smart-contract through a new transaction on the chain by adding a new *Request for countermeasures*.
3 At the time of transaction validation:
3a The validation pool will elect the nodes that will be in charge of the investigation (nodes P, C and M) and integrate the elected nodes into the replication process.
3b Before starting the replication process for this new transaction.
4 The elected nodes will then retrieve the data from the Oracle and Beacon chains that preceded the event to start searching for patterns that could have predicted the anomaly. Once the various events have been evaluated, each node will then test the modification of each of the authorized parameters *Scope of countermeasures proposals* to extract the most effective ones before measuring their impact from the locally hosted transactions (sandbox).
5 The evaluation completed, each of the nodes elected for the investigation will forward to the smart-contract storage pool *Uniris Prediction Module - 4* the most relevant proposals (Patterns / Countermeasures and KPI).
6 Once the quorum of responses received has been reached, the smart-contract storage pool *Uniris Prediction Module - 4* will locally test the different proposals and if a consensus is reached, generate an update of the smart-contract through a new transaction containing the validated countermeasure. This countermeasure will be tested again by the validation pool of this new transaction before being replicated throughout the network.
7 Once the chain is updated, each of the network nodes will integrate this new trigger into the local trigger database *Triggers Ledger* (1.3.5) to be checked during each validation, replication or self-repair process.

1.7 3.6 Billion Times Less Energy Consumption

Despite the strength of decentralized networks, based on proof of work, to migrate from trust imposed by a third party to proven and distributed trust. The energy consumption of the HashRate based Proof of Work (up to 400 kWh for the validation of a single transaction) is clearly a barrier to its widespread adoption. Even in the medium term, this massive energy consumption prevents large-scale use of this technological revolution, both for its profitability (the minimum cost of electricity consumed for a miner being 2$ per transaction) and for the number of power plants that would have to be deployed.

Other solutions have emerged such as Proof-of-Stake, but those pose an obvious problem of centralization and governance or DAG (Directed Acyclic Graph), which poses the problem of data availability.
Uniris network stands out due to its relentless focus on data availability and on the inclusive, global and decentralized spirit of proof of work (while not being based on HashRate).

The calculation below, although theoretical, is based on the first results obtained and the assumptions considered are as follows:

- Daily Consumption of a Uniris node working at 100% : 15Wh (e.g. Intel NUC i3)
- In a most conservative hypothesis for sake of calculation, consider ten nodes are dedicated for 10 seconds to validate and replicate a transaction (the current figures for validation of a transaction are less than 1 second for a single node and around 100 ms for the replication).
- The number of transactions on the Bitcoin network is 93 million transactions per year with an energy consumption of around 38.7 TWh/year.

Based on our assumption of 10 seconds per transaction validation, the number of transactions validated by these 10 nodes per year is: 365.4 × 24 × 60 × 6 = 3,157,046 transactions, and therefore the number of nodes required to handle the current yearly Bitcoin transactions would be: 93,000,000 ÷ 3,157,046 = 29.45 with sets of 10 nodes so 29.45 × 10 = 295 nodes, that is all the number of nodes required by Uniris network to cover the current mining power of the Bitcoin network.

Taking it further - In terms of energy consumption of Uniris network, we obtain 295 × (365.4 × 24) × 15 = 38,805 kWh/year (10^{-9} × 38.7 TWh/year or 3.6 Billion times less). A mere 0.42 W/sec required per transaction or the energy equivalent in joules to one tenth of a gram of sugar.

Bibliography

[1] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. *Vivaldi: A Decentralized Network Coordinate System*. SIGCOMM'04, 2004.

[2] Bernadette Charron-Bost and André Schiper. The heard-of model: Computing in distributed systems with benign failures. 01 2007.

[3] Pieter Wuille. Hierarchical deterministic wallets. 2012.

[4] Marek Palatinus, Pavol Rusnak, Aaron Voisine, and Sean Bowe. Mnemonic code for generating deterministic keys. 2013.