1) Transaction | Blocks :

The chain of blocks with transactions, is replaced with various transaction chains.

Each block is a transaction, Each transaction is a block with its own proof. There

is only one transaction in a block, with its own chain.

Each block is a transaction with its own validation evidences that will allow it

to be associated with a given chain.

2) Transaction Chains:

A transaction chain is chain of validated transactions. A a transaction is linked to the previous transaction cryptograhically, and a new transaction can only be added from the last validated transaction in the chain.

Each validated transaction is stored as a chain that can only be updated from the last validated transaction in the chain. 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 first transaction in the chain (i.e genesis) is widely used as

the reference for the chain

Transactions chains associated with network operation must be fully replicated on all nodes (Node chains, algorithms, decentralized softwares, etc.).

txn 0(genesis txn) -> txn 1 -> txn 2 -> txn 3 ... -> txn index

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 @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.

3) Different Types of Transaction Chains:

#//-Public Smart-Contracts and Decentralized Identity Ledgers

Chains( UCO wallet),

Smart-Contract Transactions Chains,

Decentralized Identities Transactions Chains for Nodes,

Transactions chains of the authorized nodes,

Beacon Chains,

Oracle Chains,

Storage Chains,

Node Chain,

Network Transactions Chains,

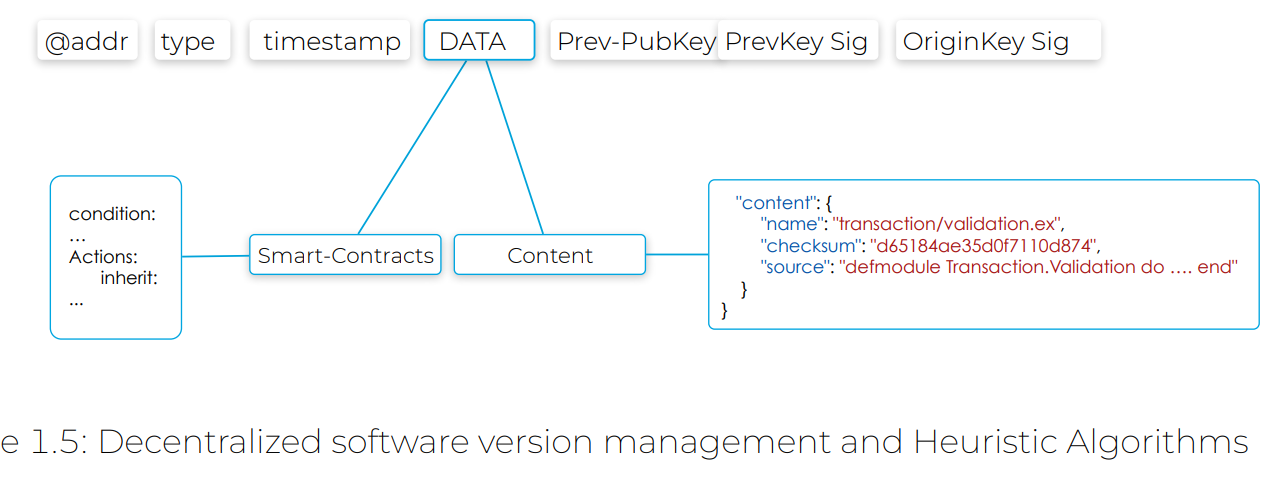
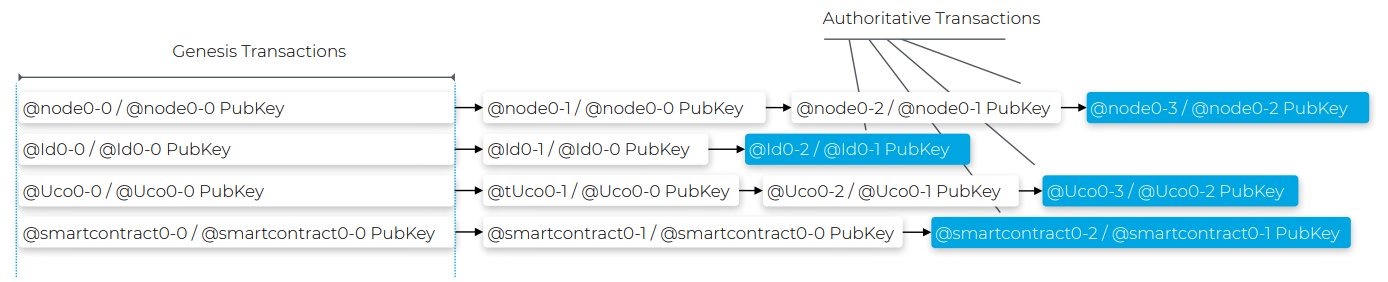
Keychains,

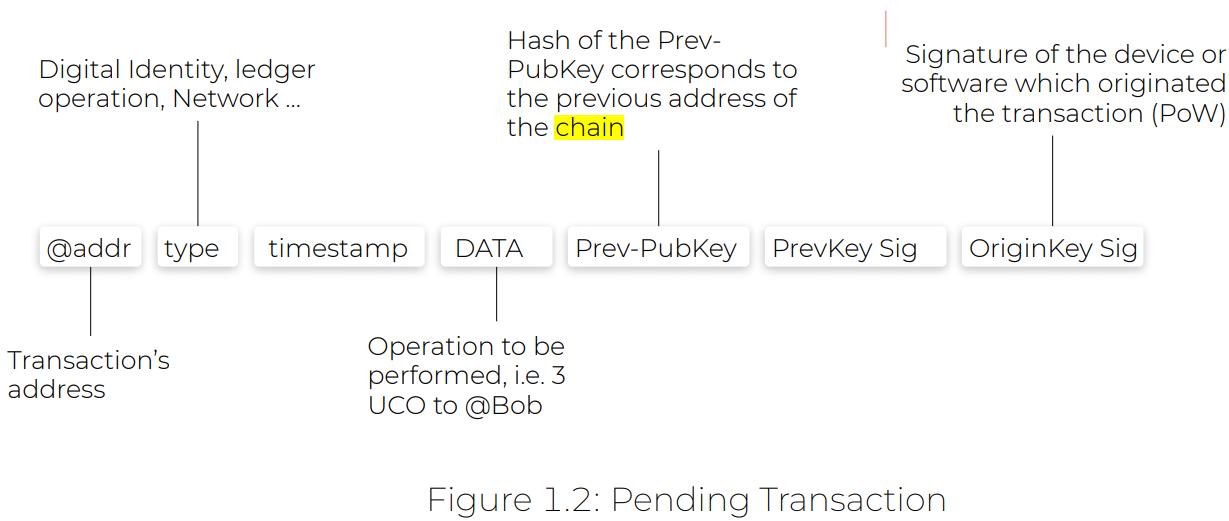
Transactions chains of Heuristic Algorithms and software,

Shared Secrets Chains of Nodes and Devices

Prediction module Chain

Lifecycle / Structure of Transactions

.



Transaction-address :

An unused address that corresponds to the hash of the public key of the transaction that will succeed it in the chain.

ORIGINKEY SIG:

IT is the signature rom the hardware , sotware device/wallet that generated the transaction Signature of the device or software which originated the transaction. 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

PRIVKEY 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")

PREV-PUBKEY: If the current transaction number is y , then the public key associated with y-1th transaction is placed here

Genesis txn address :

Transaction address is Hash of the previous public key For the 0th (genesis ) and 1st txn same public key is placed

In PREV-PUBKEY is placed and same PREVKEY SIG .

0TH

{PUBKEY0, privkey}Crypto.derive\_keypair(seed, 0 , curve)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Addr | Type | Timestamp | Data | Prev-Pubkey | PrevKeySig | OriginKEY SIG |
| Hash(PUBKEY) |  |  |  | PUBKEY |  | Sign(origin-priv-key, this txn) |

1st

Crypto.derive\_keypair(seed, 1 ,curve)

{PUBKEY, privkey}Crypto.derive\_keypair(seed, 1, curve)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Addr | Type | Timestamp | Data | Prev-Pubkey | PrevKeySig | OriginKEY SIG |
| Hash(PUBKEY) |  |  |  | PUBKEY0 |  | Sign(origin-priv-key, this txn) |

Validated Transaction

Nodes Directives

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.

4) **ARCH Consensus**

Atomic Rotating Commitment Heuristic

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.

**Atomic Commitment**

I n nodes are required to achieve consensus, then all n nodes shoud agree on a validation o a transaction, I even one nodes reues to the success of validation then it is considered maliciouis .

Heuristics: (algorithms, softwares and parameters) , 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 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

**Heuristic Algorithms/Softwares are stored in a decentralized way as transaction chains or more precisely as smart-contract chains**

**Global, Unpredictable and Reproducible Election**

To achieve:

unpredictable, global but locally executed, verifiable and reproducible election of transaction validation nodes

* Hash of the transaction content
* The daily nonce from the shared secrets of nodes which is renewed daily.
* the last public key of the node chain Last Node Pubkey
* the computation of the Validation Rotating Keys

**RotatingKeyNodePubkey = Hash(Last Node Pubkey, Daily Nonce, hash(transaction content))**

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