Distributed Ledger Technologies Applications

Sistemi Peer-to-Peer
2019/2020

Mirko Zichichi
mirko.zichichi2@unibo.it

CONTENTS OF THIS LECTURE

1. IOTA Applications
    - Masked Authenticated Messaging
    - Data Marketplace
2. State Channels
    - Ethereum µRaiden
    - Flash Channels
    - Ethereum Plasma
3. Introduction to
    - IPFS
    - zk-SNARK

IOTA

Second Layer Applications

IOTA Recap

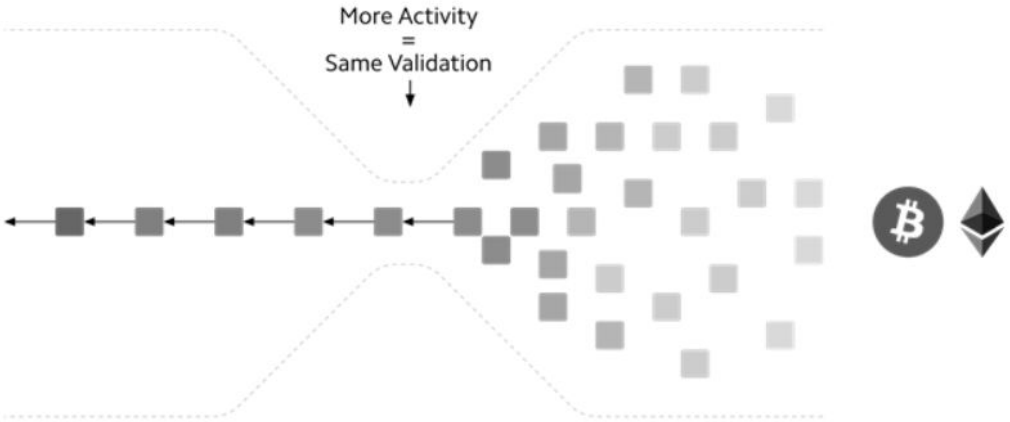
To solve some inefficiencies of the blockchain, IOTA is based based on a new concept of DLT, allowing the link between **Internet of Everything (IoE)** and Web 3.0.

- With the rapid development of IoT industry, it is increasingly important to allow **machine-to-machine micropayments** efficiently.

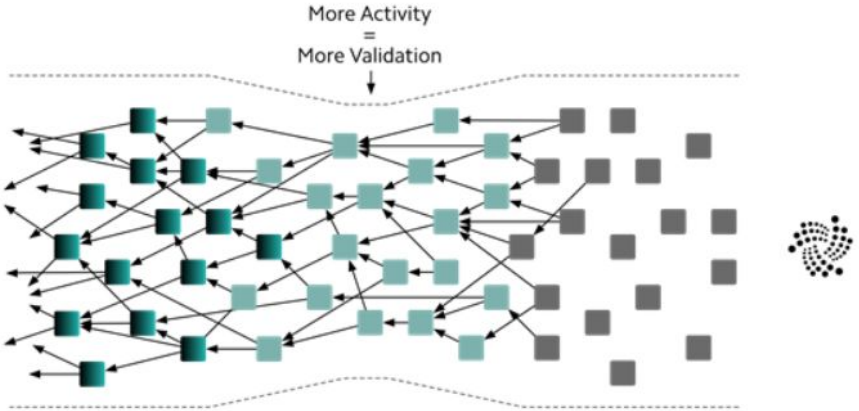
- In the case of Ethereum, for instance, it is possible that a **transaction fee** is larger than the amount of value being transferred but, it is not easy to get rid of fees in the blockchain since they serve as an incentive for the creators of blocks.

- To address this issue IOTA introduced the **Tangle**.

THE BLOCKCHAIN BOTTLENECK



THE IOTA TANGLE SCALES!



https://www.switchmycrypto.com/blog/iota-miota-simple-explanation/

IOTA Structure

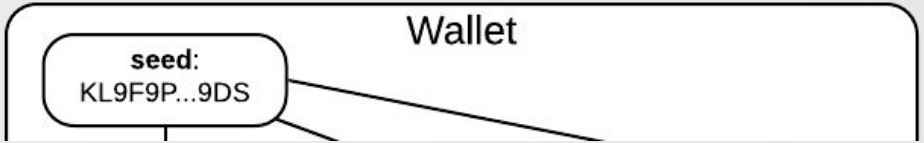
- There are no Miners, therefore no miner fees and no incentives to slow the network down to raise fees and other conflicts of interest.

- An IOTA network consists of **IRI nodes** (full nodes) and **clients**.
  - **IRI Node**: device that has read/write access to the Tangle
  - **Client**: device that has a seed.

- A **seed** gives a client access to his addresses and these have a balance
- In order to operate, clients must send **bundles of transactions** to a node so that the nodes can validate the transactions and update their ledgers.

IOTA Structure
Wallet

KL9F9PVPUGHHKEKKDSFFEYYTVSFRXOUWFH9LZIKKKQ
EDO9L9MK9LIVOZUIPCML9RCHNDR QYPNGNOU9DS

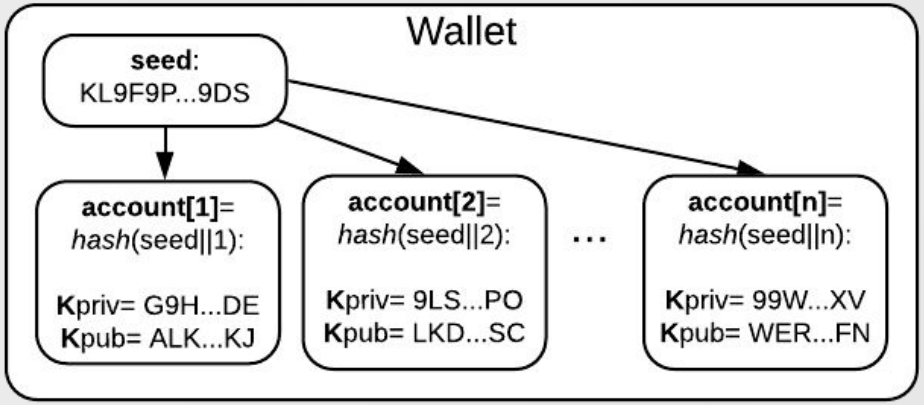


- **Trytes**: IOTA represents data according to the trinary numeric system.
  All data represented in IOTA is tryte-encoded, according to the tryte alphabet which contain all the English alphabet letters plus the character '9'

IOTA Structure
Wallet

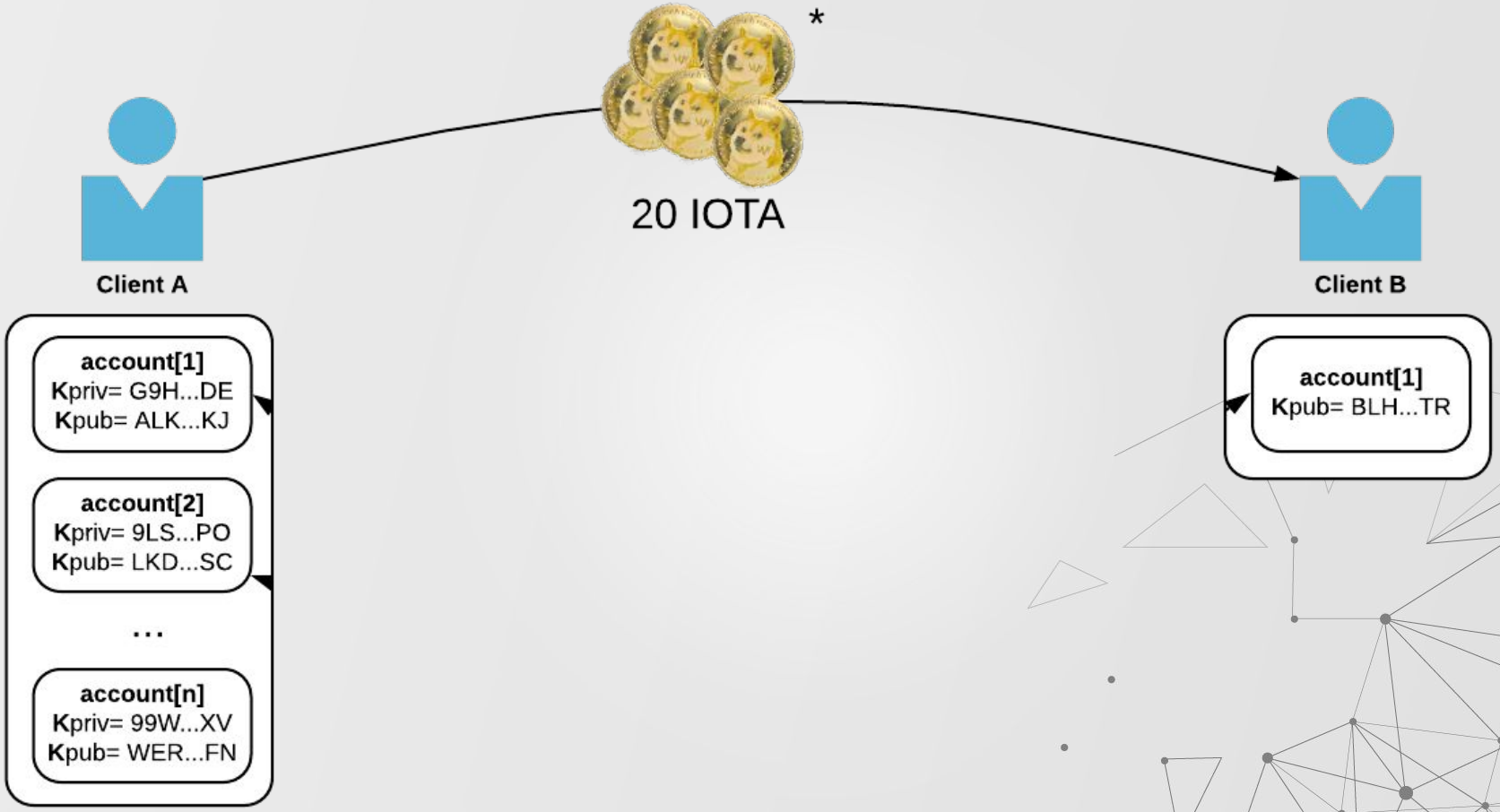
Client A

KL9F9PVPUGHHKEKKDSFFEYYTVSFRXOUWFH9LZIKKKQ
EDO9L9MK9LIVOZUIPCML9RCHNDR QYPNGNOU9DS



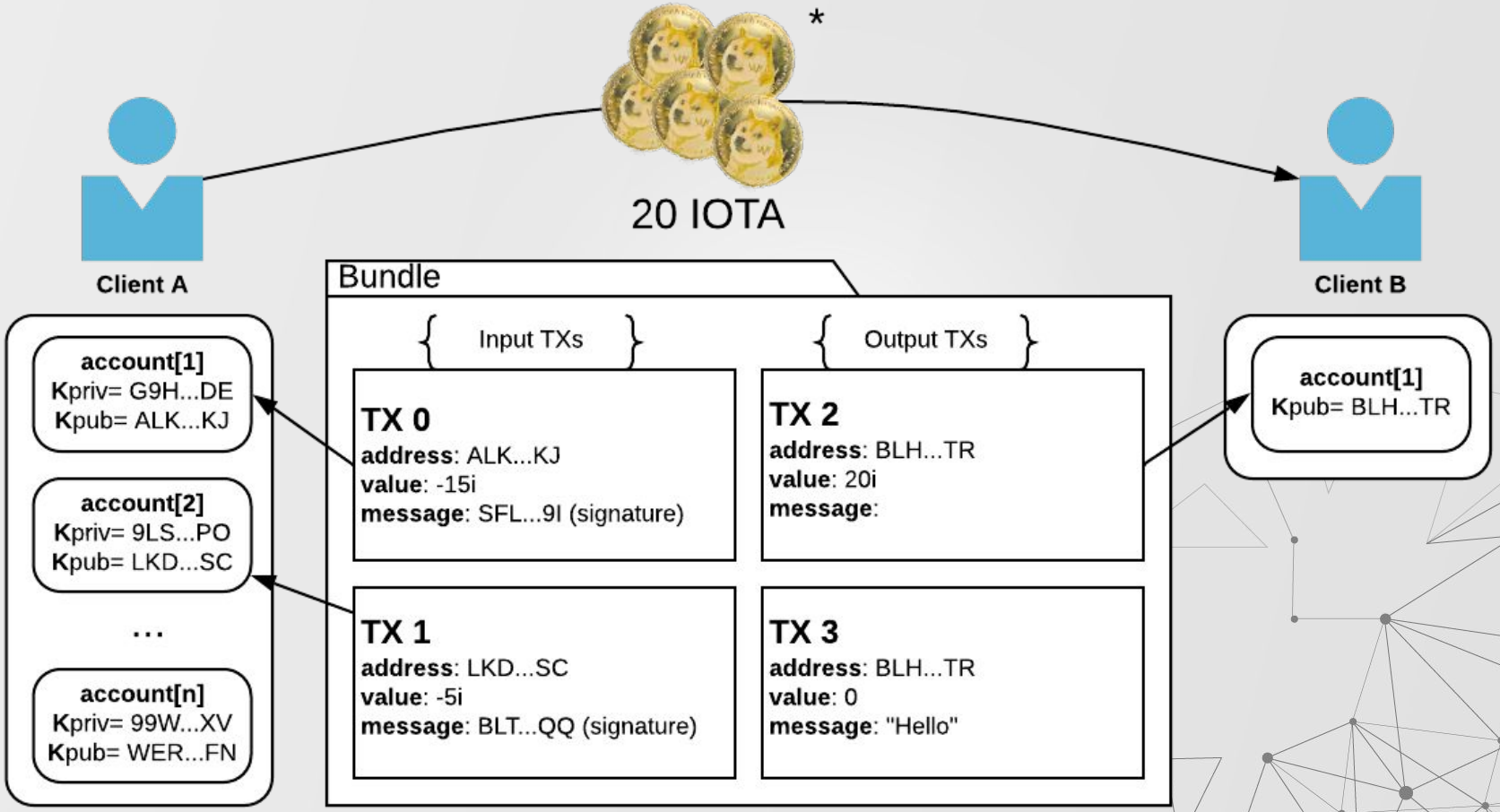
IOTA Structure
Bundle & Transactions

* https://dogecoin.com/

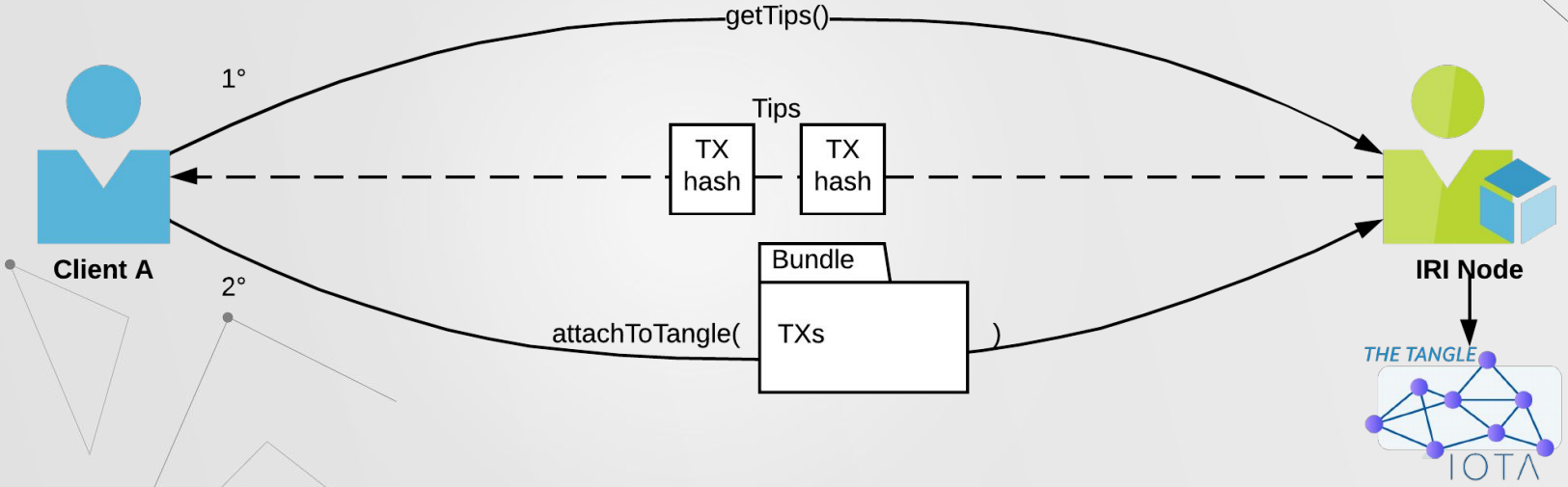


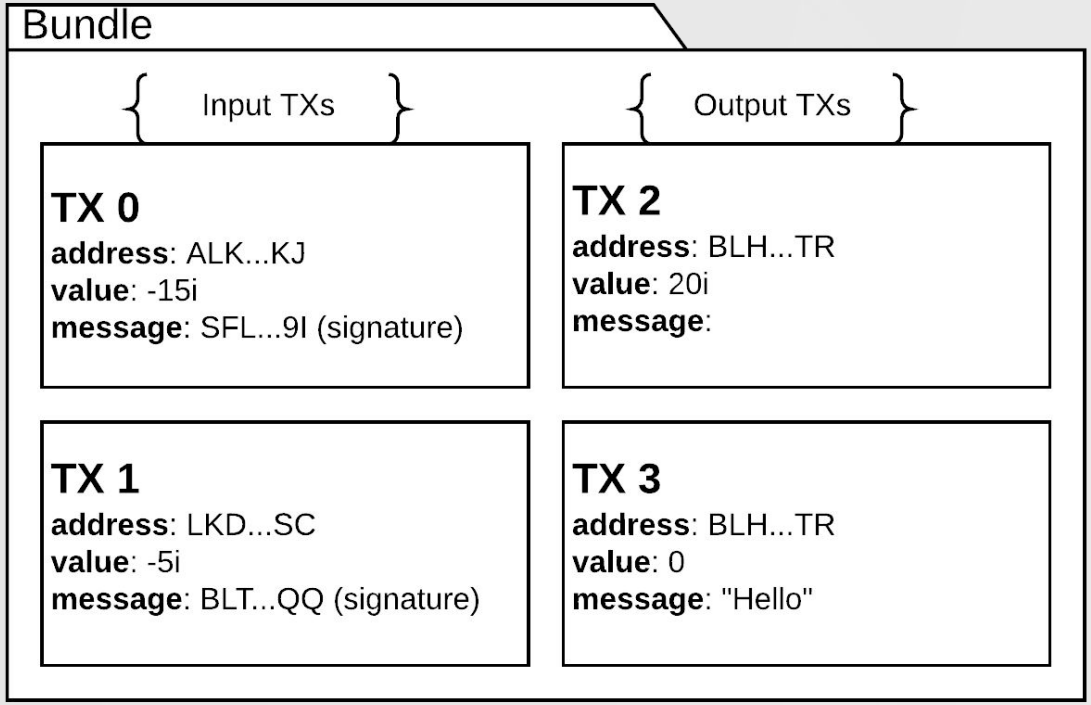
IOTA Structure
Bundle & Transactions

* https://dogecoin.com/

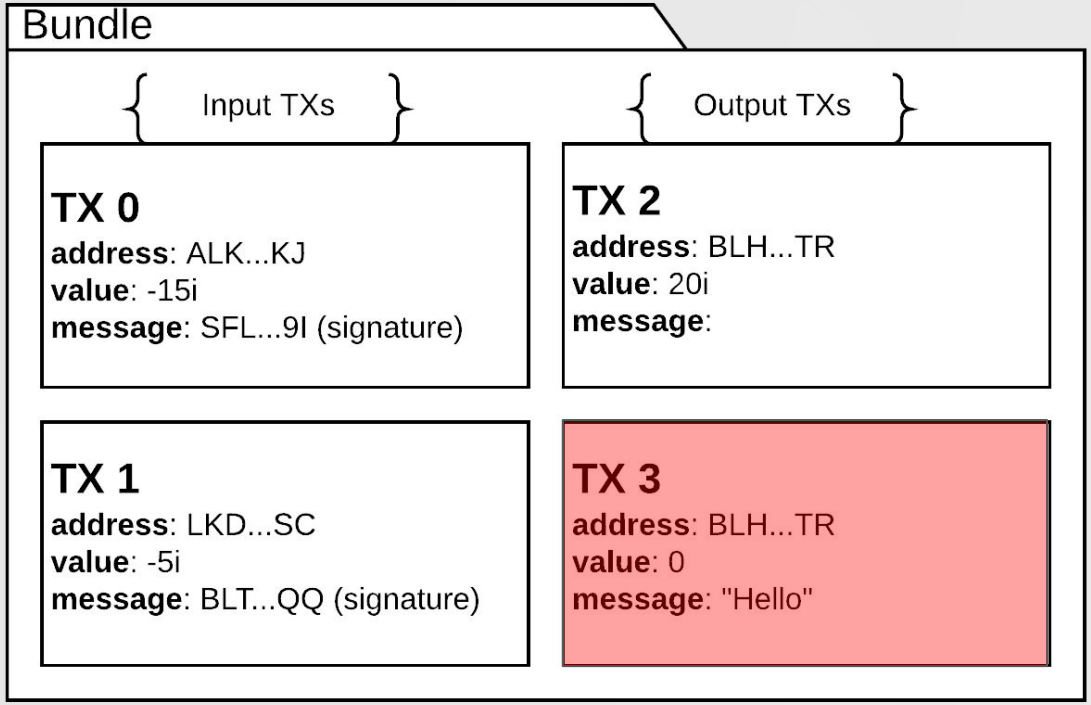


IOTA Structure
IRI Node




Bundle
Input TXs
Output TXs
TX 0
address: ALK...KJ
value: -15i
message: SFL...9I (signature)
TX 2
address: BLH...TR
value: 20i
message:
TX 1
address: LKD...SC
value: -5i
message: BLT...QQ (signature)
TX 3
address: BLH...TR
value: 0
message: "Hello"

Messaging


Bundle
Input TXs
Output TXs
TX 0
address: ALK...KJ
value: -15i
message: SFL...9I (signature)
TX 2
address: BLH...TR
value: 20i
message:
TX 1
address: LKD...SC
value: -5i
message: BLT...QQ (signature)
TX 3
address: BLH...TR
value: 0
message: "Hello"

Masked Authenticated Messaging

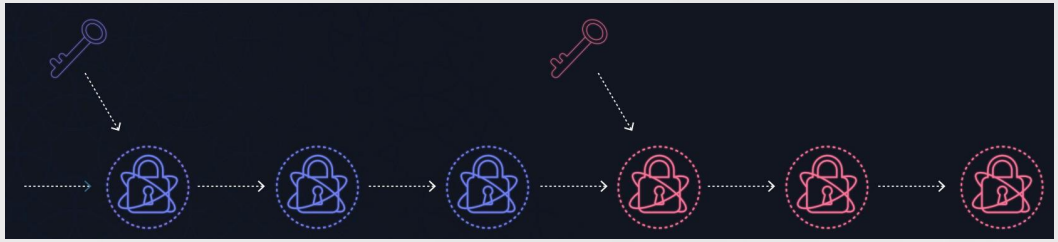
Masked Authenticated Messaging

Masked Authenticated Messaging (**MAM**) is a second layer data communication protocol which adds functionality to emit and access an encrypted data stream over the Tangle. MAM allows to maintain **integrity** and **confidentiality** in a **data** flow produced by a device

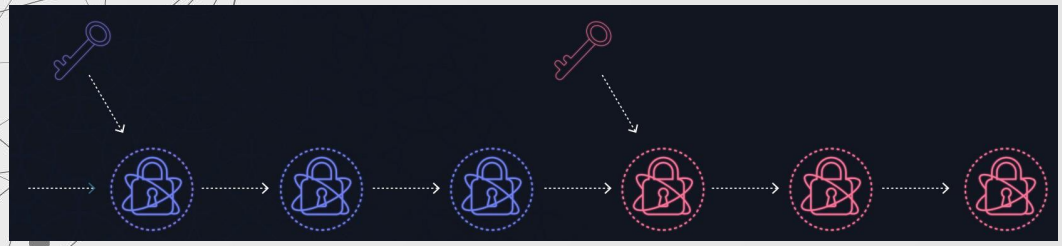
- Works through **Channels**, where publisher and viewers meet
- Viewers **subscribe** to channels to get the data that channel owner **publishes**
- This **ownership** is implemented and secured through the owner seed

MAM Channels

Channels can be considered as message chains where each message (bundle) is linked to the next one. Message address is obtained from a generated parameter called **root**



https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e

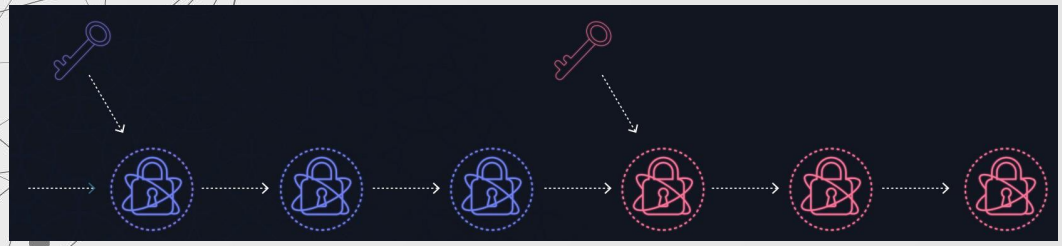


MAM Channels

Channels can be considered as message chains where each message (bundle) is linked to the next one. Message address is obtained from a generated parameter called **root**

- **Public** - *root* is used as the address of the (zero value) transaction that the message is published to (and also to encrypt the data). Anyone that knows a message *root* will be able to access all messages published from that one on.

https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e

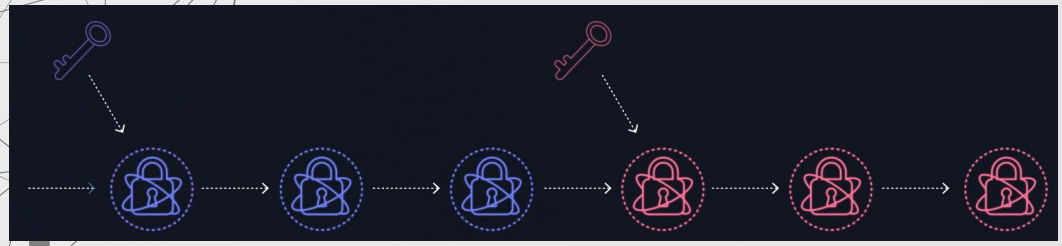


MAM Channels

Channels can be considered as message chains where each message (bundle) is linked to the next one. Message address is obtained from a generated parameter called **root**

- **Public** - *root* is used as the address of the (zero value) transaction that the message is published to (and also to encrypt the data). Anyone that knows a message *root* will be able to access all messages published from that one on.
- **Private** - *root hash* is used as the messages address and *root* to encrypt the message. Knowing the address it is not possible to derive the *root* --> cannot decrypt the message

https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e



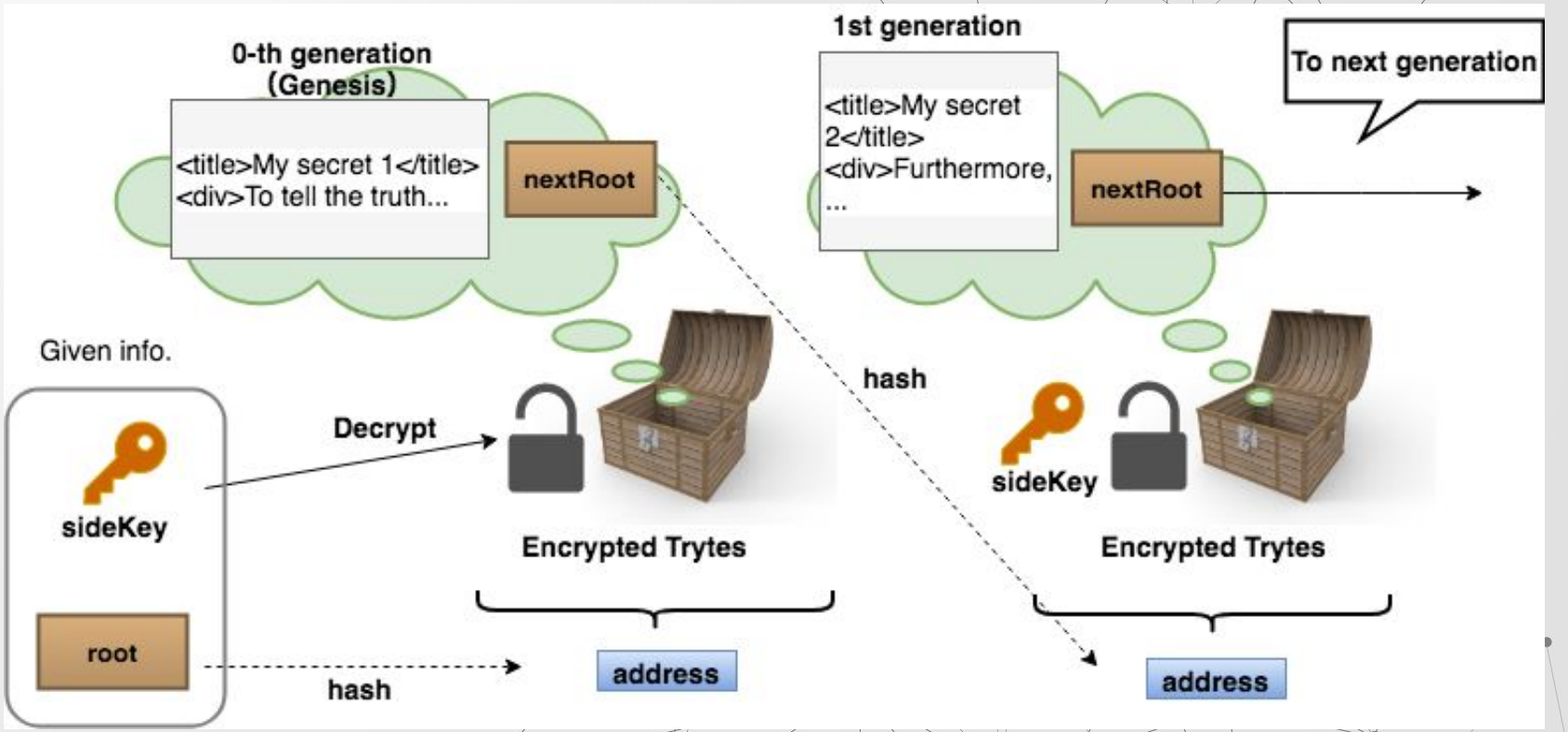
MAM Channels

Channels can be considered as message chains where each message (bundle) is linked to the next one. Message address is obtained from a generated parameter called **root**

- **Public** - *root* is used as the address of the (zero value) transaction that the message is published to (and also to encrypt the data). Anyone that knows a message *root* will be able to access all messages published from that one on.
- **Private** - *root hash* is used as the messages address and *root* to encrypt the message. Knowing the address it is not possible to derive the *root* --> cannot decrypt the message
- **Restricted** - *root* is combined with an *authorization key* and the hash is used as the messages address. The *side key* is also used to encrypt the data.
  The publisher could stop using the side key without changing their channel, so access could be revoked to subscribers

https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e

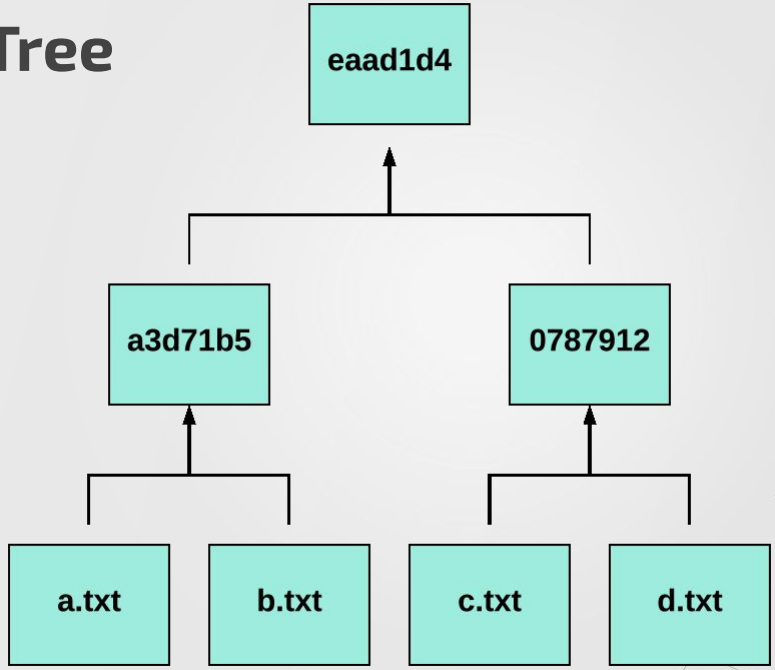
MAM
Restricted Channel



https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413

MAM
Where to get the ROOT?

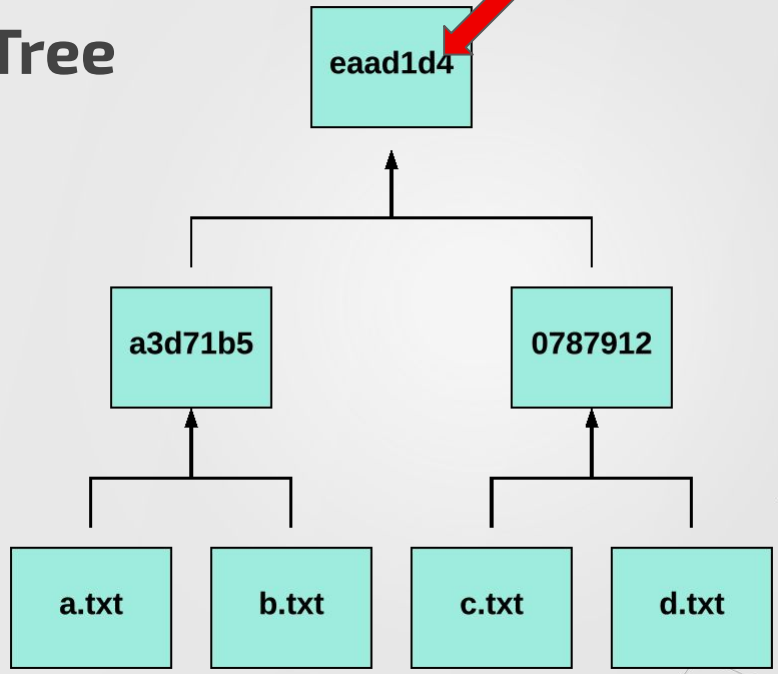
Merkle Tree



https://medium.com/byzantine-studio/blockchain-fundamentals-what-is-a-merkle-tree-d44c529391d7

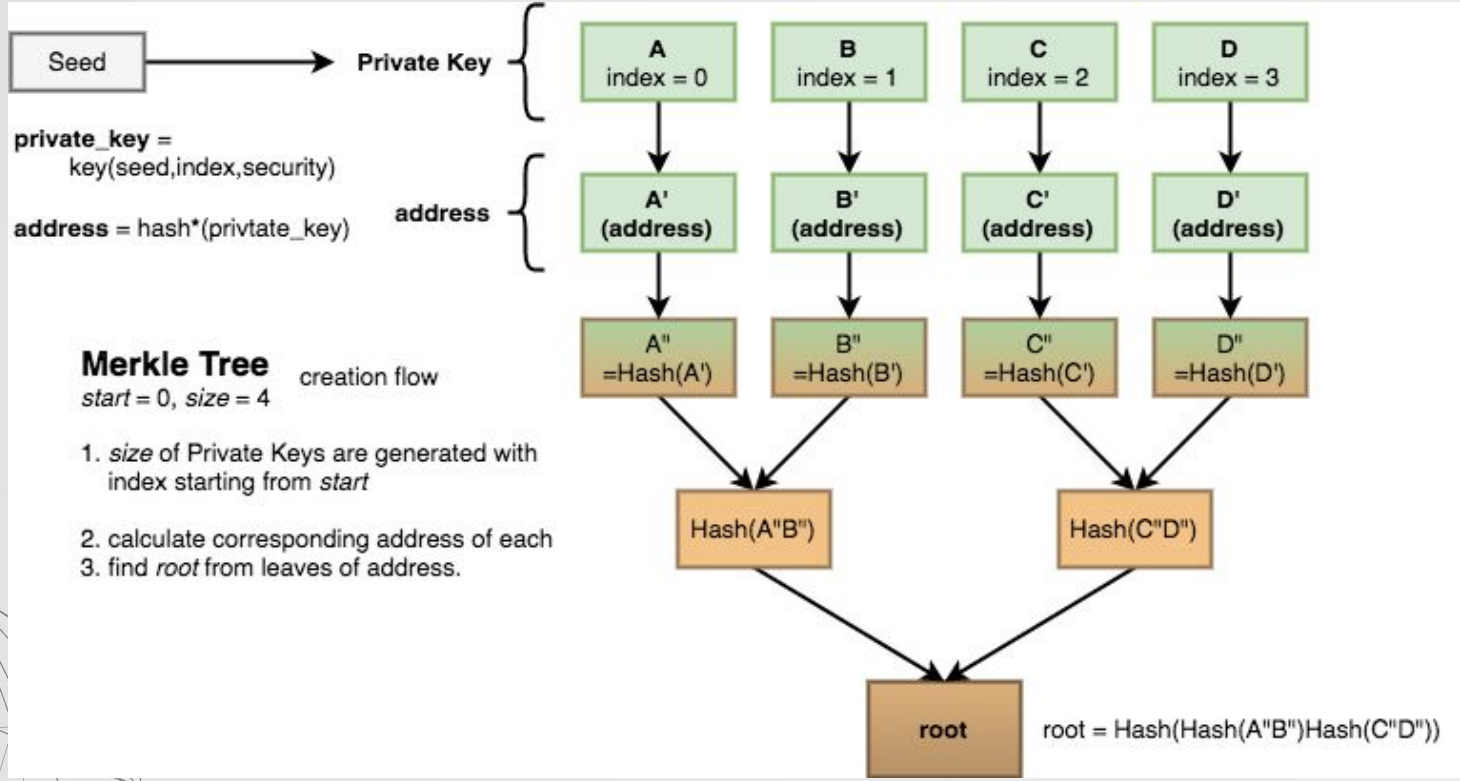
MAM
Where to get the ROOT?

Merkle Tree



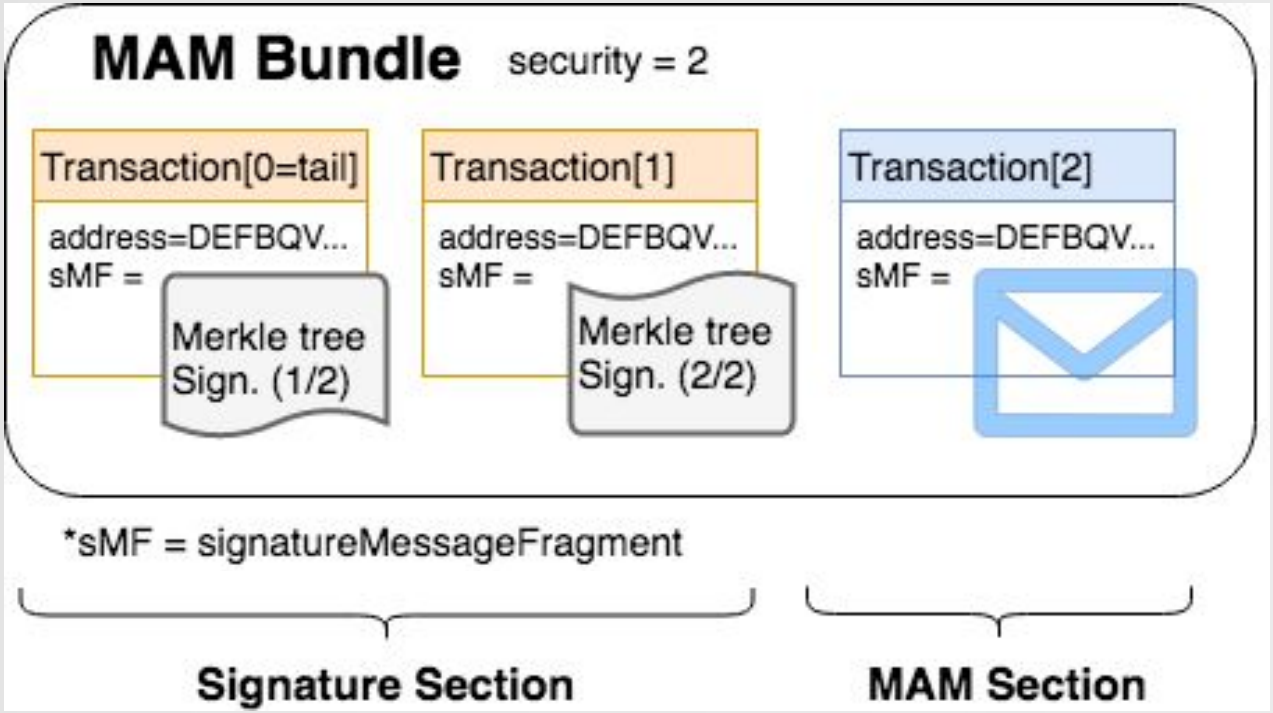
https://medium.com/byzantine-studio/blockchain-fundamentals-what-is-a-merkle-tree-d44c529391d7

MAM
Merkle tree based signature scheme



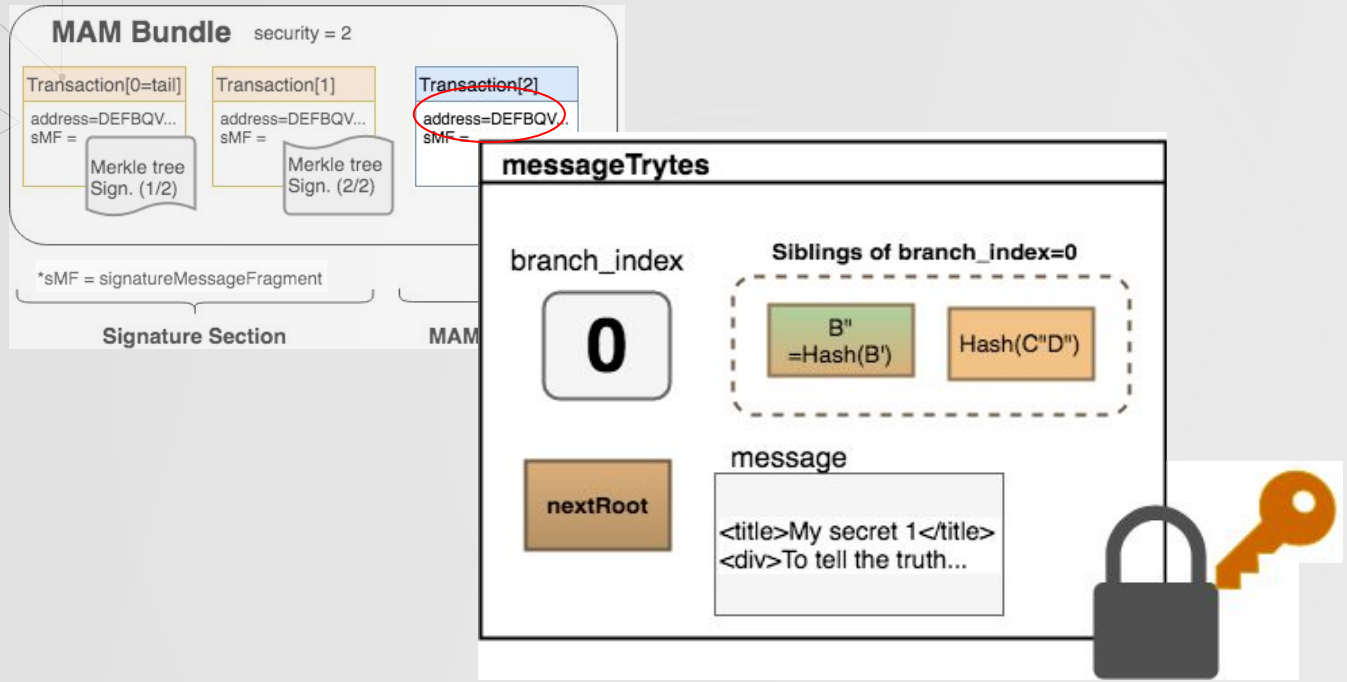
https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413

MAM
Bundle Structure



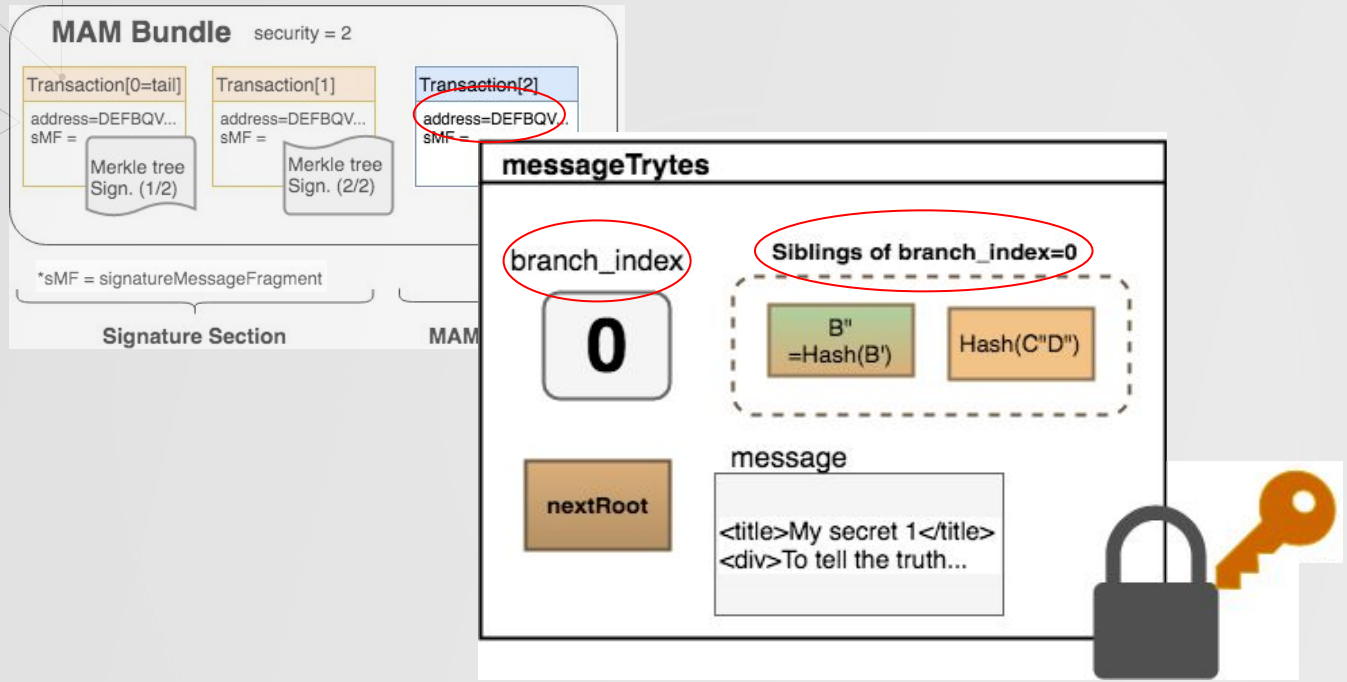
https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413

MAM
Bundle Structure



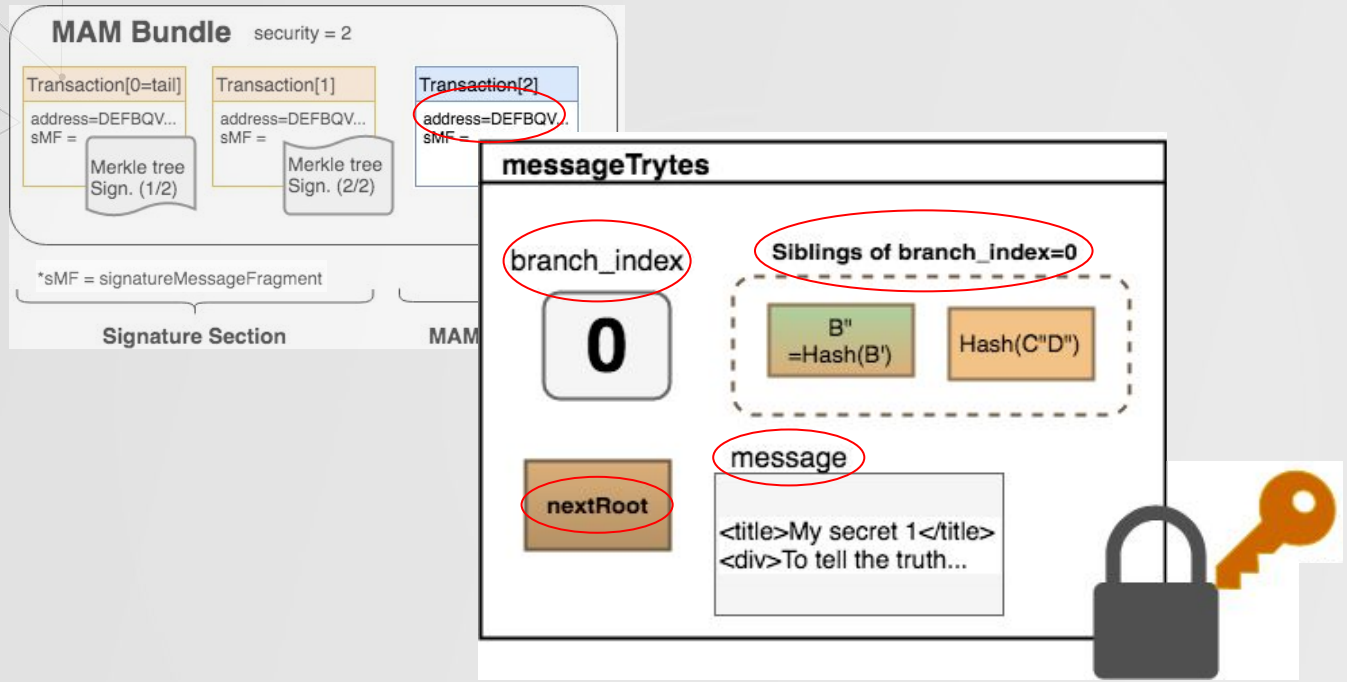
https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413

MAM
Bundle Structure



https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413

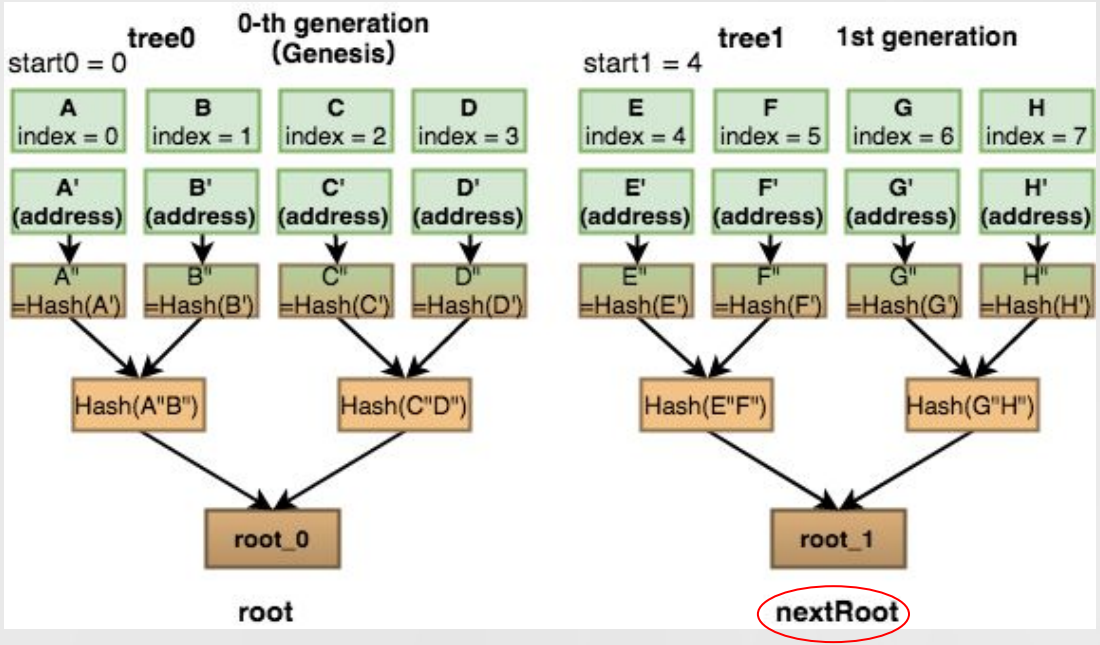
MAM
Bundle Structure



https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413

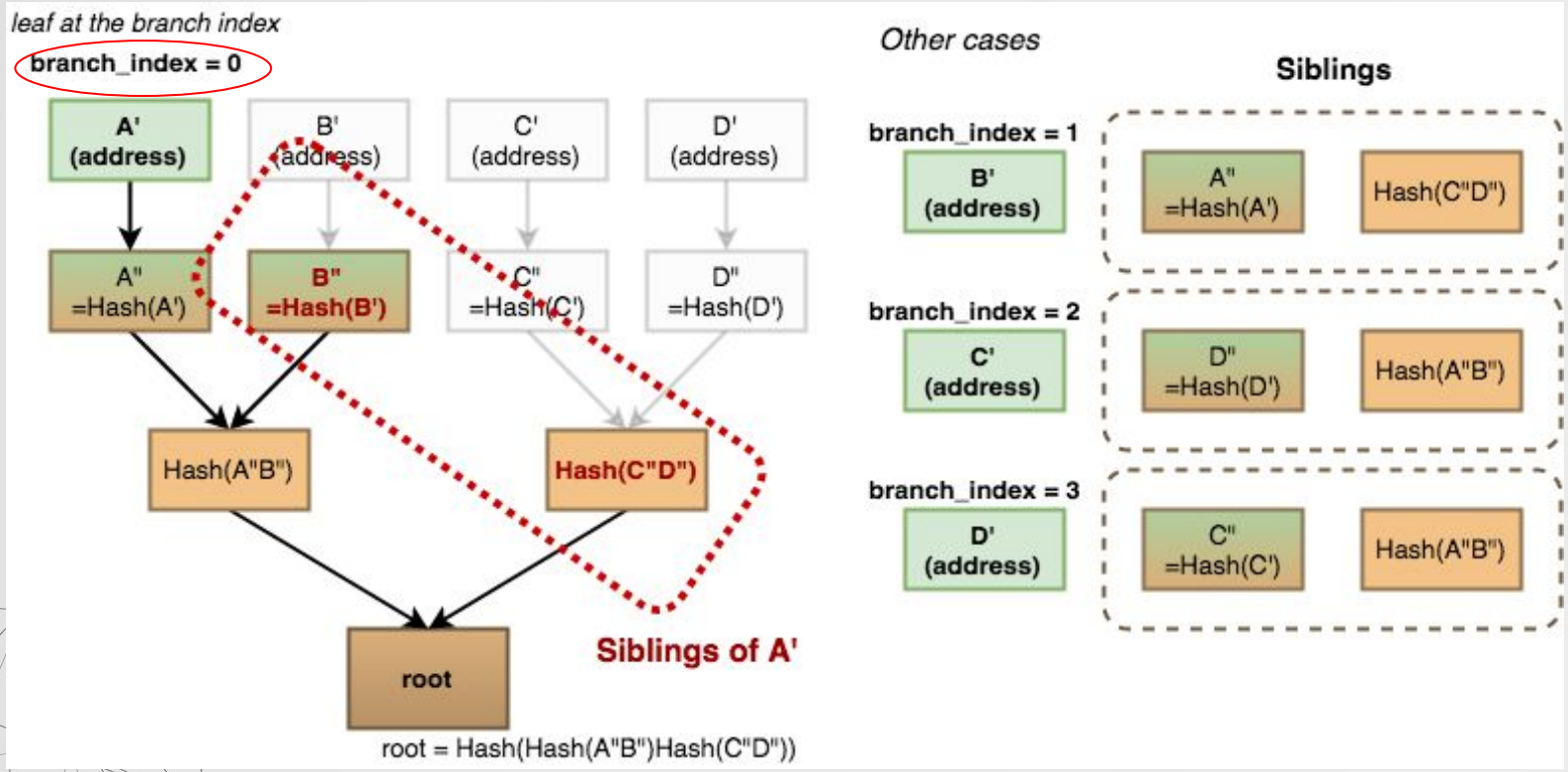
MAM
Bundle Structure- Next Root

To post masked message of one generation, one must generate TWO merkle trees.
First tree is for the current generation and the other is for next generation.



https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413

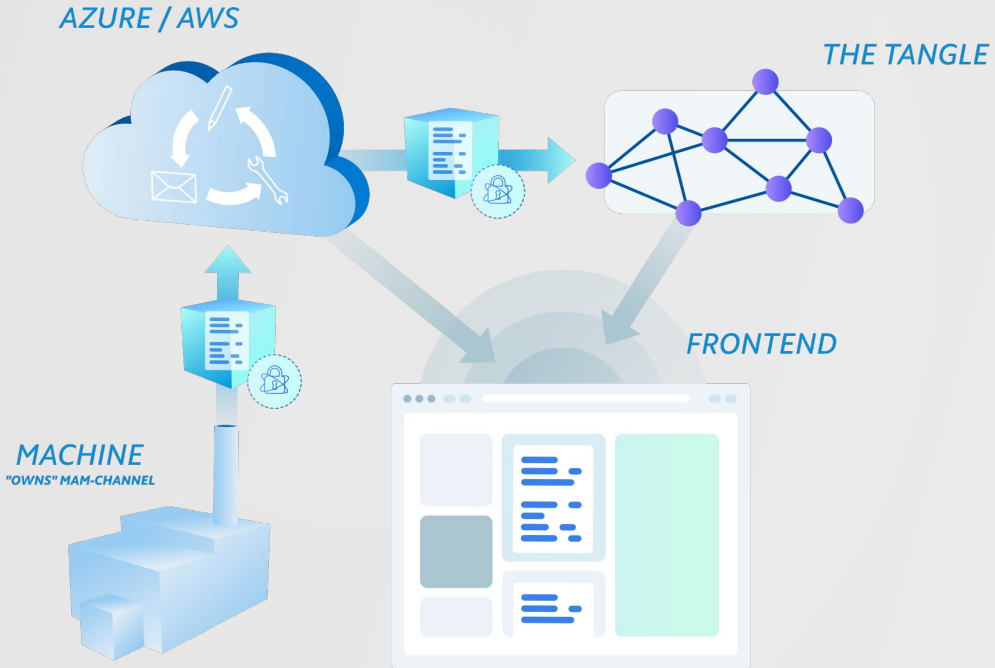
MAM
Bundle Structure - Branch Index and Siblings



https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413

Data Marketplace

Launched by the IOTA Foundation in 2017, as a Proof of Concept and open innovation ecosystem. Nowadays it is a system that makes it possible to securely **store, sell** and **access** data streams.



https://data.iota.org/#/specs

Data Marketplace

- **Masked Authenticated Messaging**
  In the Data Marketplace implementation MAM is used and each data packet is encrypted with it is own randomly generated encryption key. All keys are securely stored in a cloud backend system.

- **Devnet**
  As well as other DLTs, IOTA also has a separate ledger used for testing.
  It allows developers to work on their apps without costing them any real tokens.

- **Payment**
  Data can be decrypted if payment is made. Price of the data stream is defined by sensor owner, and usually set between 1000 and 50000 IOTAs. But it is important to notice that IOTA Token in the Devnet are worth nothing, thus buyer can fund his wallet for free and spend tokens exclusively for purchasing sensor data.

Data Marketplace

- **Data storage in the Cloud**
  The Data Marketplace PoC uses a cloud backend service provided by Google Firebase. All information is stored securely and confidentially. Google cloud services are used for user authentication and access rights management.

- **Data Streaming**
  New data packets can be added with a recommended minimum time interval of 5-10 min. More frequent additions are possible, but since a publisher needs to conduct a small amount of Proof of Work to allow the data to propagate through the network, the delay could be up to 60 seconds for every new packet.

STATE CHANNELS

Payment channels off-chain

State Channels (or Micropayments)

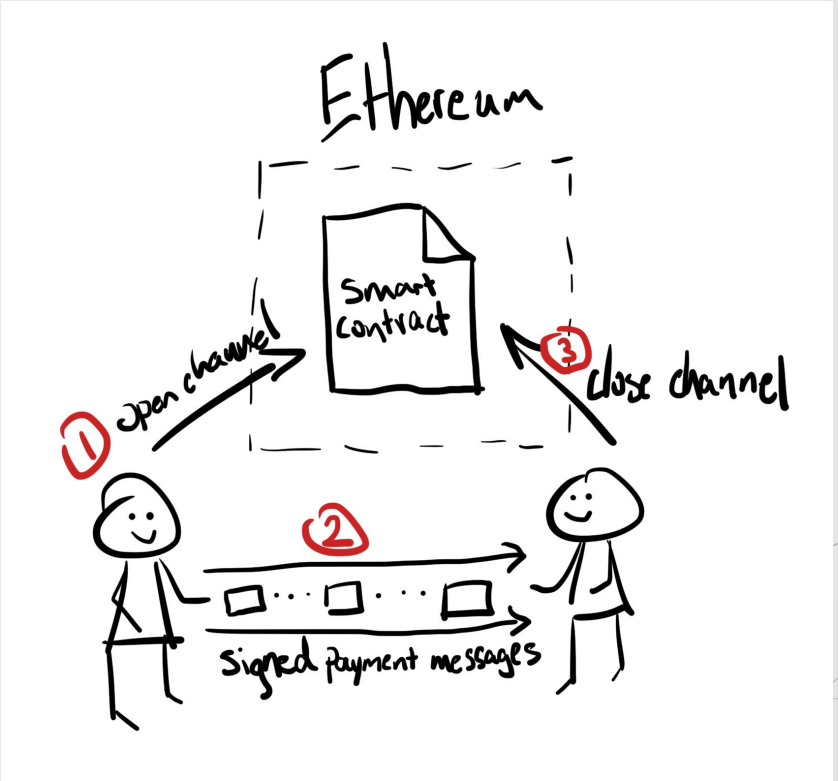
State Channels are a design pattern for instant blockchain transactions made **off-chain**. As transactions do not touch the blockchain, **fees** and **waiting times** are avoided in a secure way.

State Channels (or Micropayments)

State Channels are a design pattern for instant blockchain transactions made **off-chain**. As transactions do not touch the blockchain, **fees** and **waiting times** are avoided in a secure way.

These allows **micropayments** that are not registered directly on the blockchain. After an **agreement** made between two parties (e.g. through a smart contract), the two can start **exchanging messages** that indicates the current **balance** of currency among the two.

State Channels
Ethereum Smart Contract

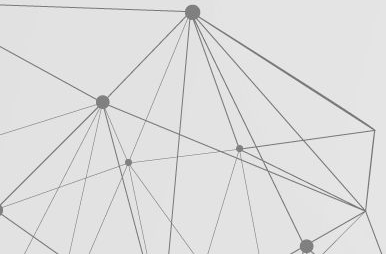


https://programtheblockchain.com/posts/2018/02/23/writing-a-simple-payment-channel/

https://microraiden.readthedocs.io/en/latest/

State Channels
µRaiden

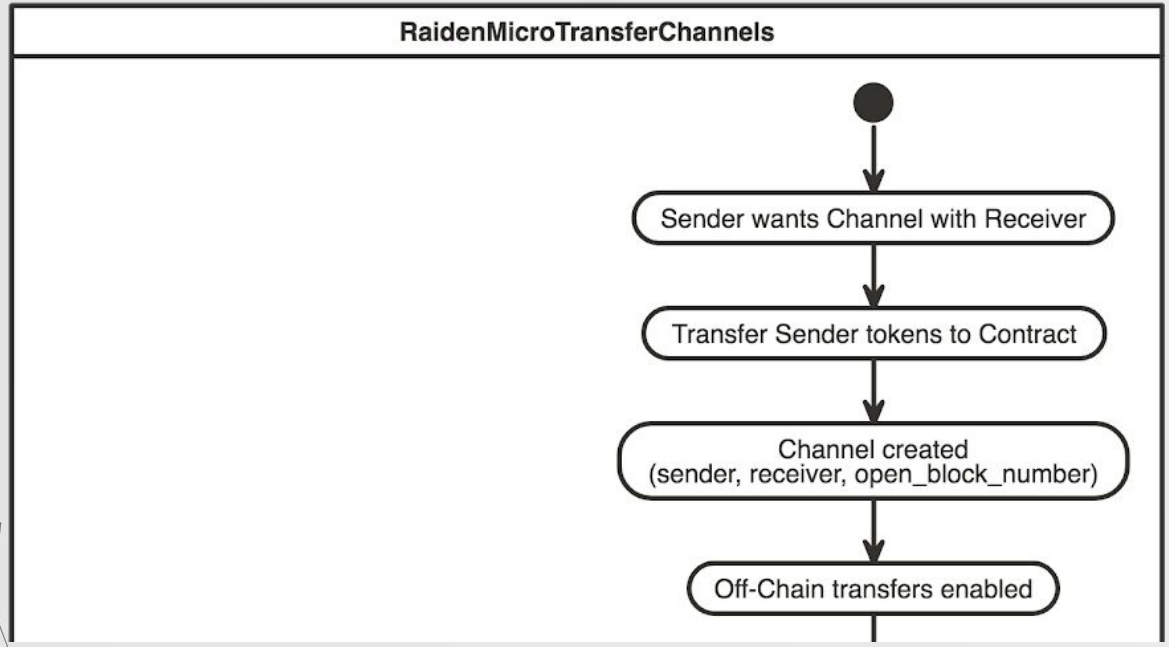
Payment channel **framework** for frequent, fast and free **ERC20** token based micropayments between two parties. Set of **open source libraries**, documentation, and code examples for multiple use cases, ready to implement on the Ethereum mainnet.



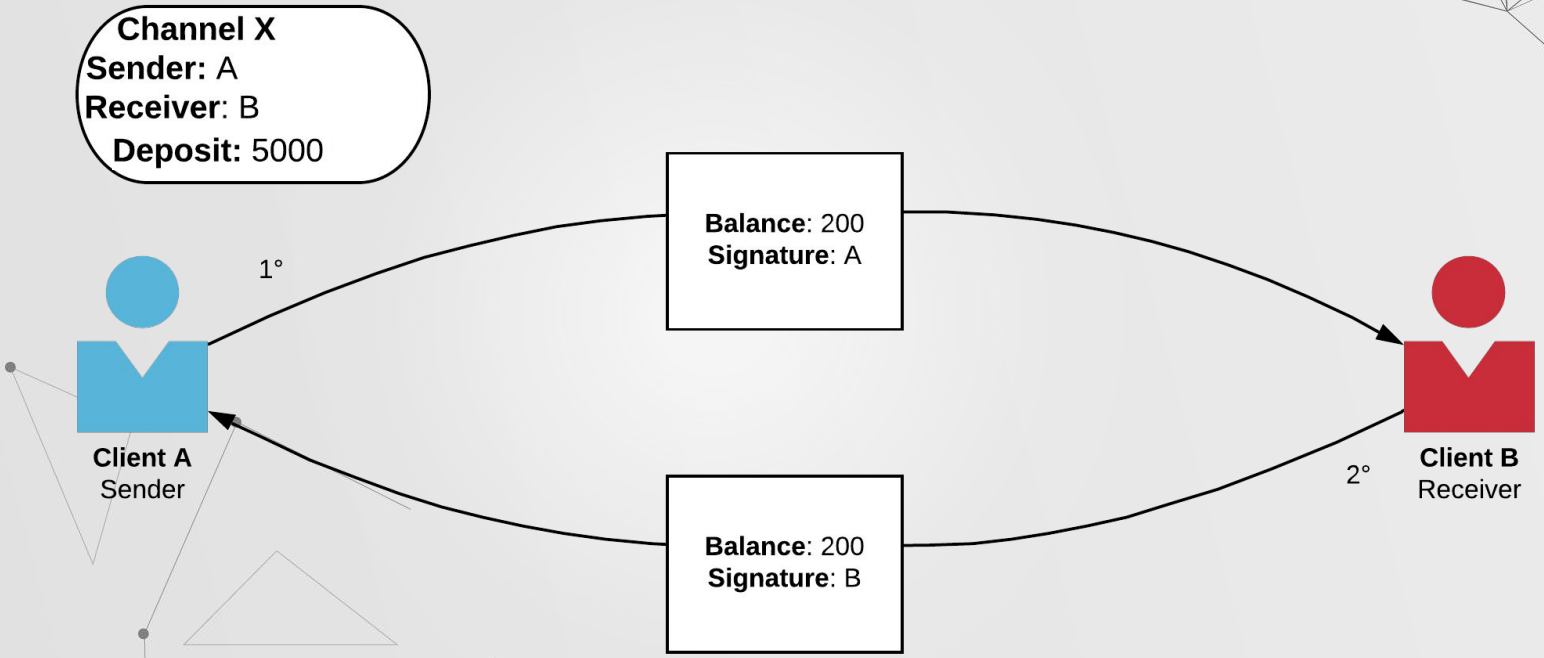
https://microraiden.readthedocs.io/en/latest/

State Channels
µRaiden

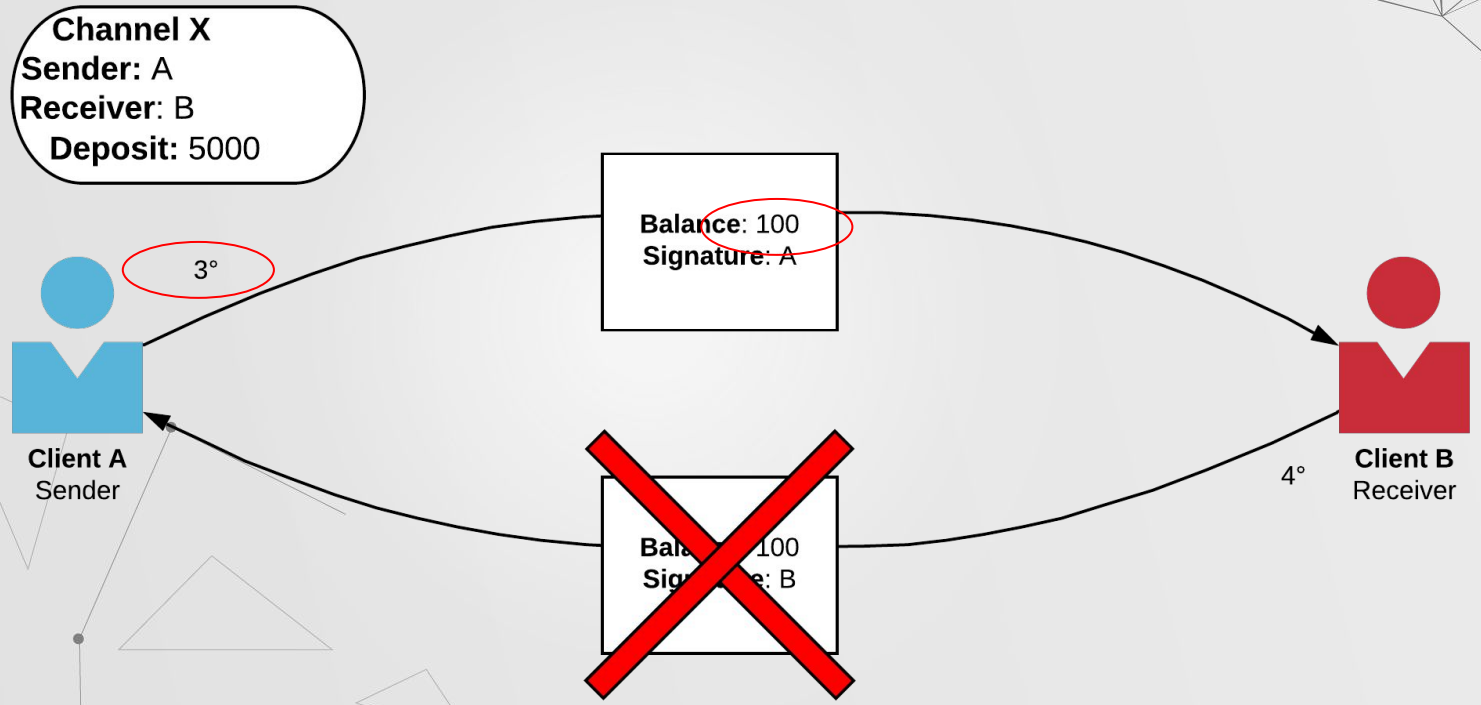
Payment channel **framework** for frequent, fast and free **ERC20** token based micropayments between two parties. Set of **open source libraries**, documentation, and code examples for multiple use cases, ready to implement on the Ethereum mainnet.



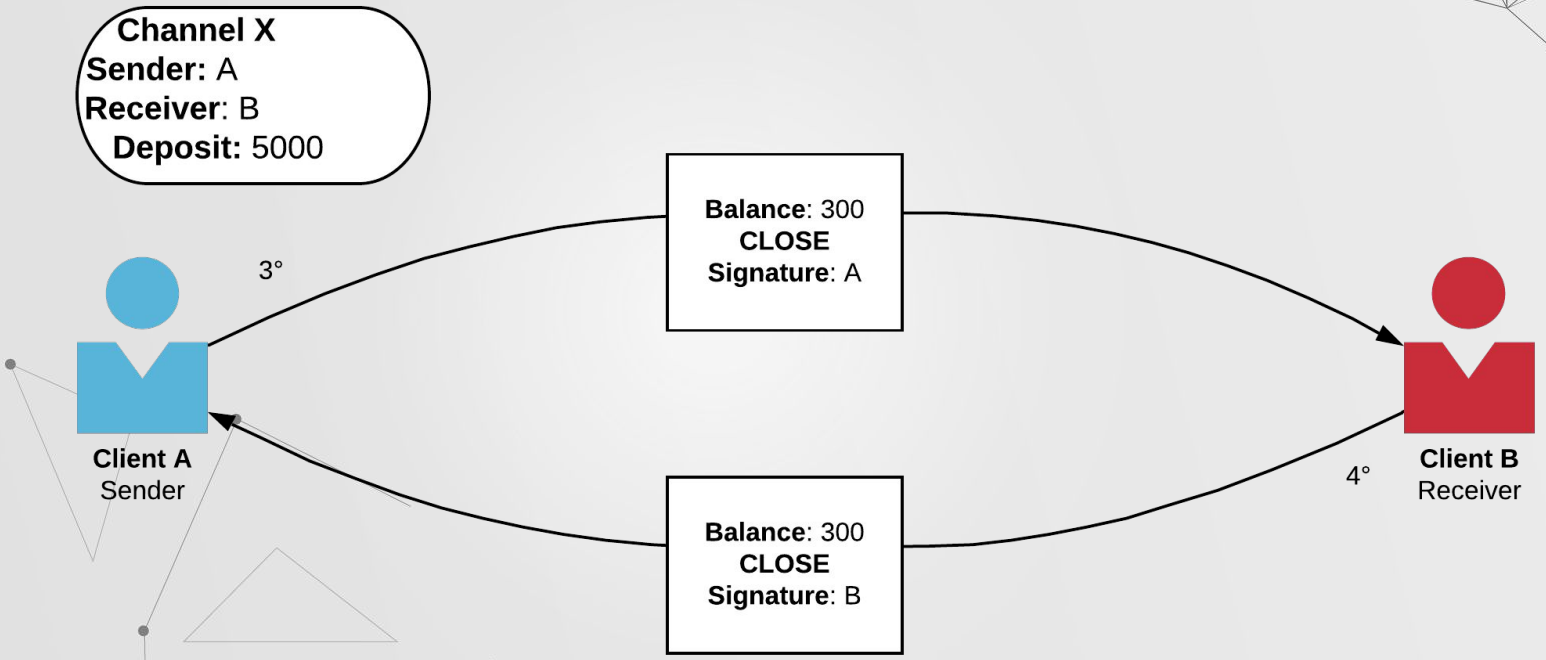
State Channels
Off-Chain Transfer



State Channels
Off-Chain Transfer

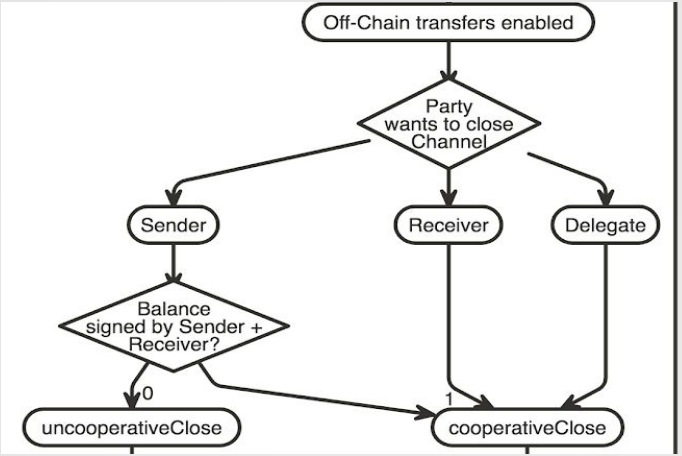


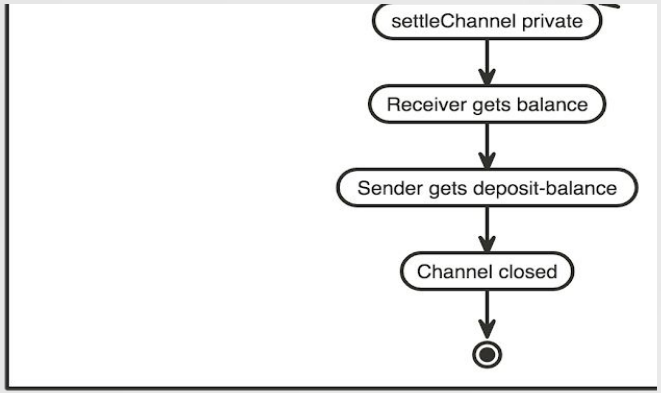
State Channels
Off-Chain Transfer



https://microraiden.readthedocs.io/en/latest/

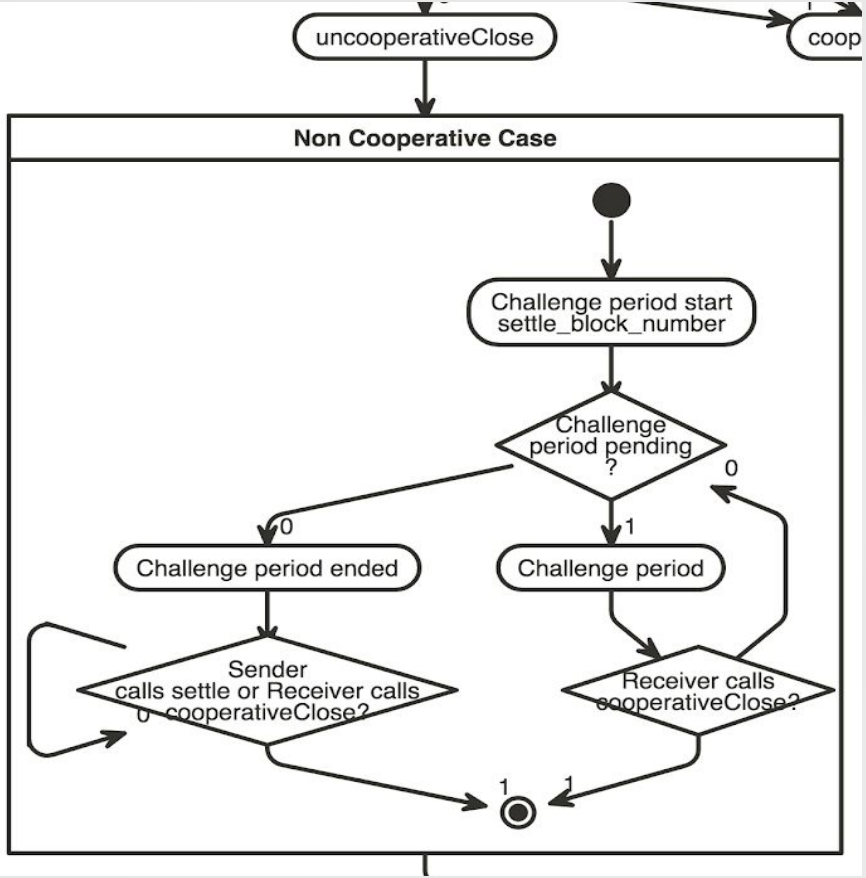
State Channels
µRaiden - Cooperative Close





https://microraiden.readthedocs.io/en/latest/

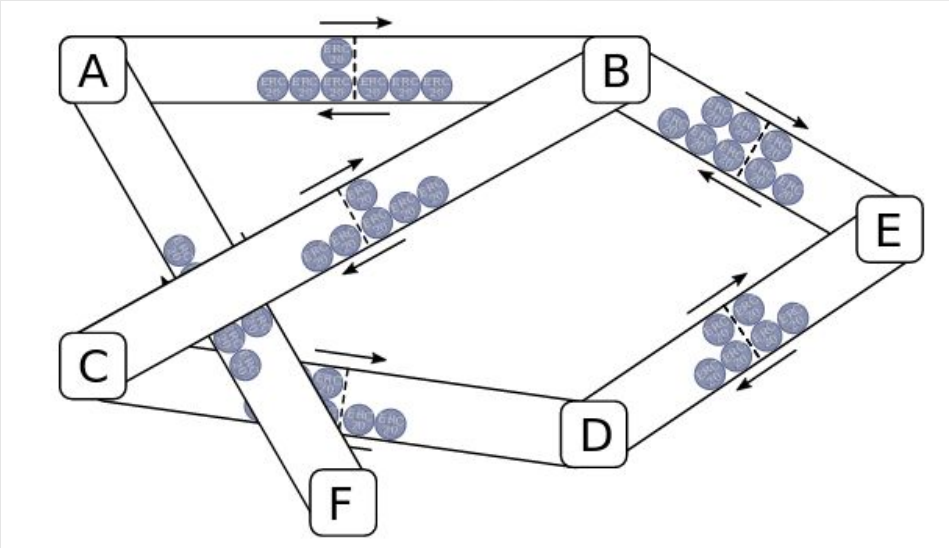
State Channels
µRaiden - Uncooperative Close



https://raiden.network/101.html

State Channels
Ethereum Raiden Network
(Bitcoin Lightning Network)

Payment channel transfers do not require any fees. **Intermediaries** within the greater network, however, will want to charge fees on a low percentage basis for **providing their own channels** to the network, leading to complex routing and a competitive channel fee **market**.



https://blog.iota.org/instant-feeless-flash-channels-88572d9a4385

State Channels
IOTA Flash Channels

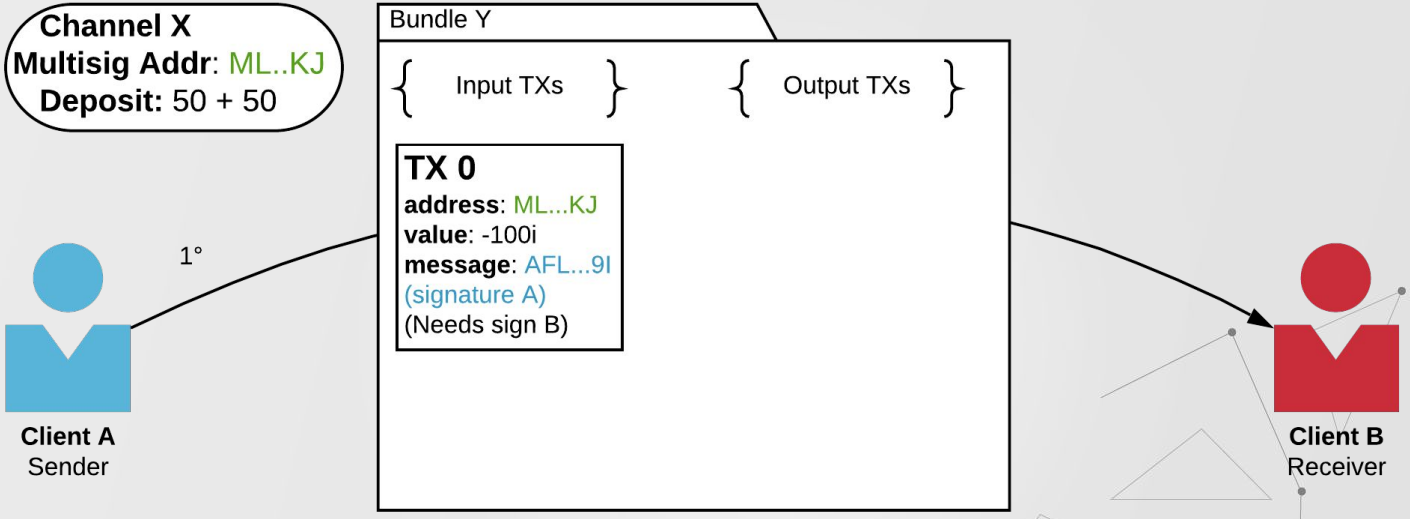
Flash is a bi-directional **off-Tangle** payment channel working in a similar fashion compared to the blockchain solutions.

When a channel is created each party **deposits** an equal amount of IOTA into a *__multi-signature address__ controlled by all parties.

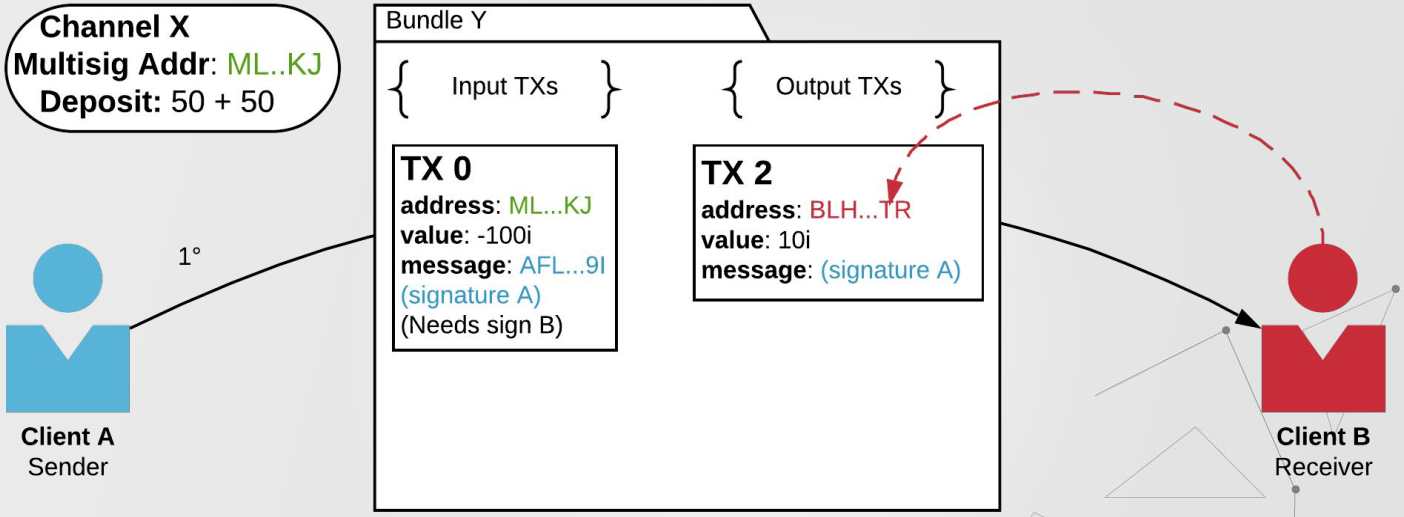
Incentives are required, like with any multi-signature scenario, as a signing party may **refuse to continue** signing transactions.

* https://medium.com/@abmushi/iota-signature-and-validation-b95b3f9ec534

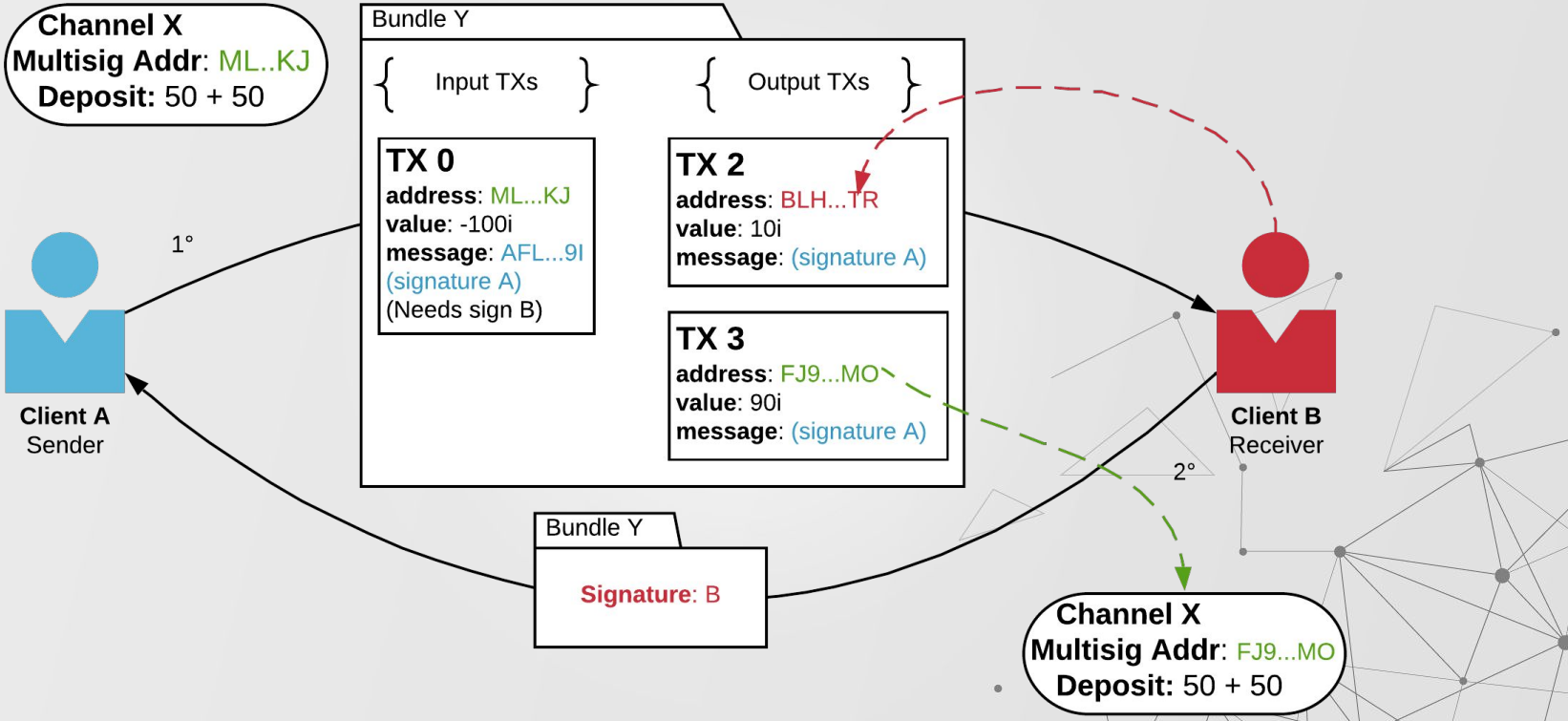
State Channels
Flash Channels - Off-Tangle Transaction



State Channels
Flash Channels - Off-Tangle Transaction



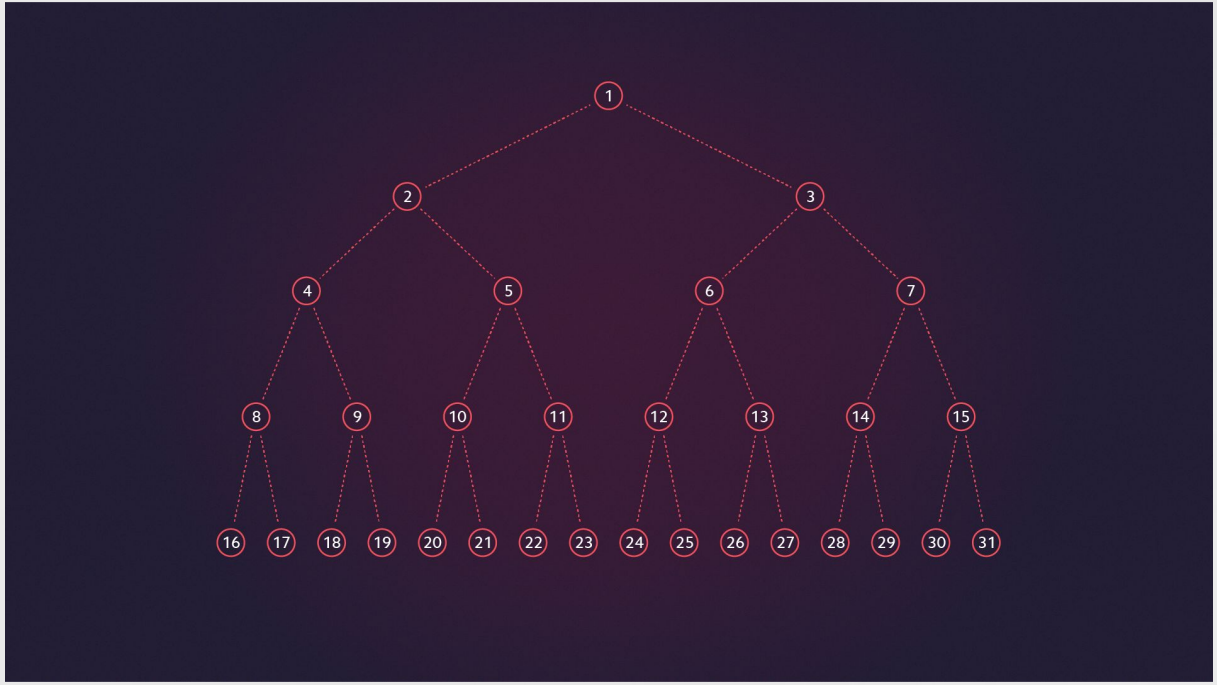
State Channels
Flash Channels - Off-Tangle Transaction



https://blog.iota.org/instant-feeless-flash-channels-88572d9a4385

State Channels
Flash Channels - Private Keys

An address can be used twice while remaining reasonably secure



For example: a 60 transaction channel requires a tree with a depth 6 as log2(60) = 5.907

Plasma (Ethereum)

Plasma is a **design pattern** that allows for off-chain messages to dictate the transfer of on-chain assets.

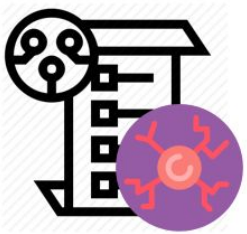
Each **Plasma chain** condenses messages about transaction ordering into a **single hash stored on the root chain**. Bitcoin and Ethereum are examples of root chains.

https://medium.com/crypto-economics/what-is-plasma-plasma-cash-6fbbef784a

Plasma

1: Wannabe operator deploys Plasma contract to the mainnet

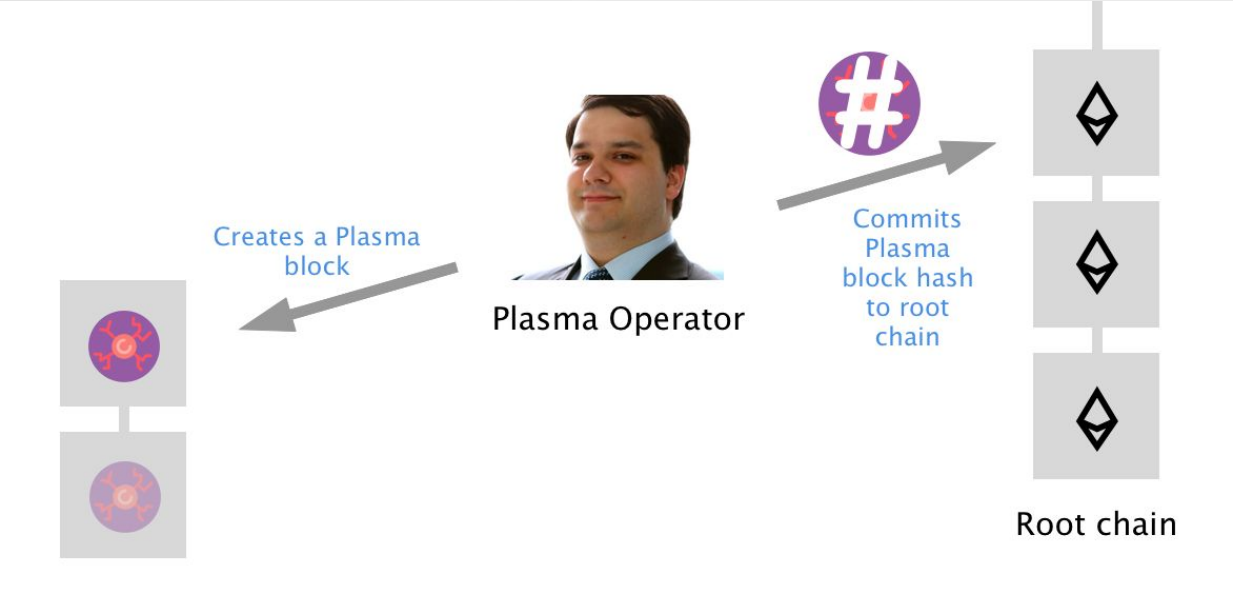
Wannabe Plasma Operator



Deploys Plasma smart contract to the mainnet

Plasma

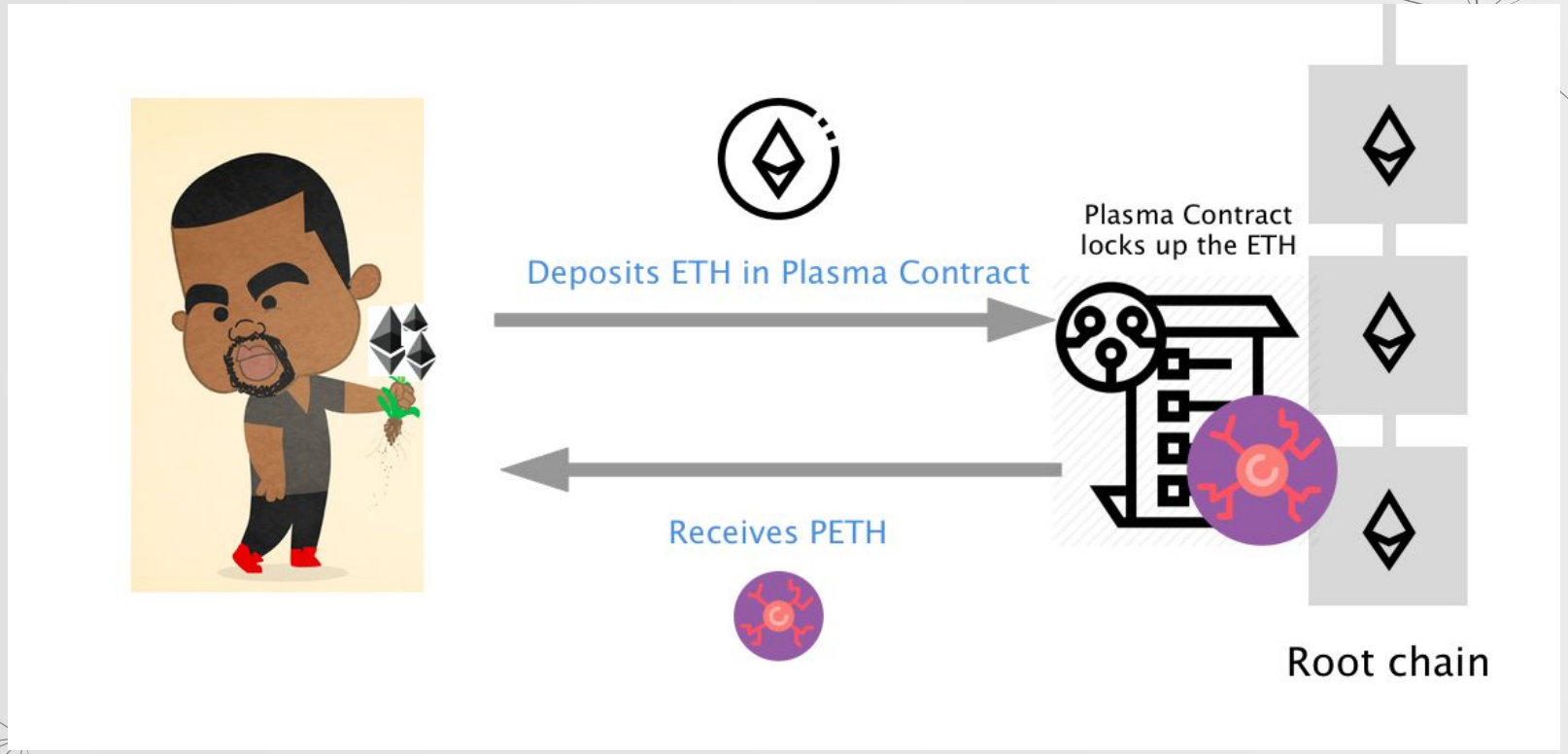
2: Plasma operator creates a block



Different **Plasma chains** can have different governance **rules**, different **tokens**, methods of **storing state**, etc.

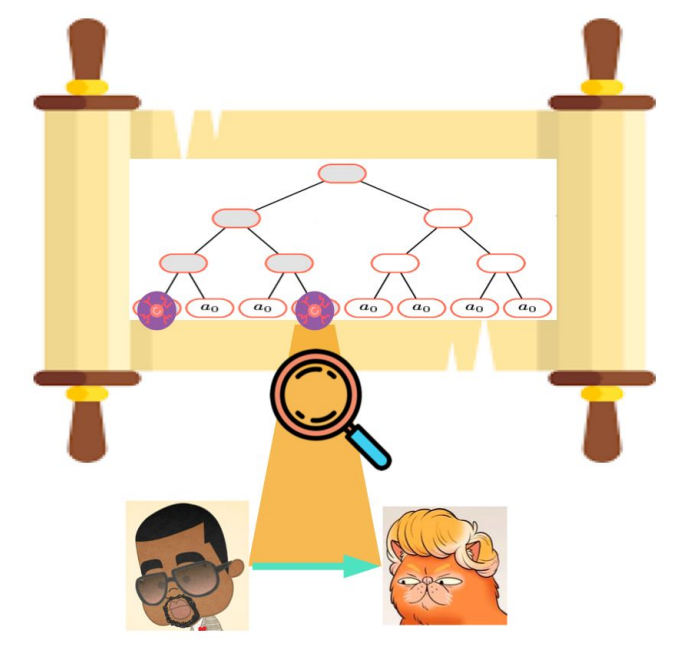
Plasma

3: Kanye deposits ETH into the Plasma contract and is assigned PETH in return



Plasma

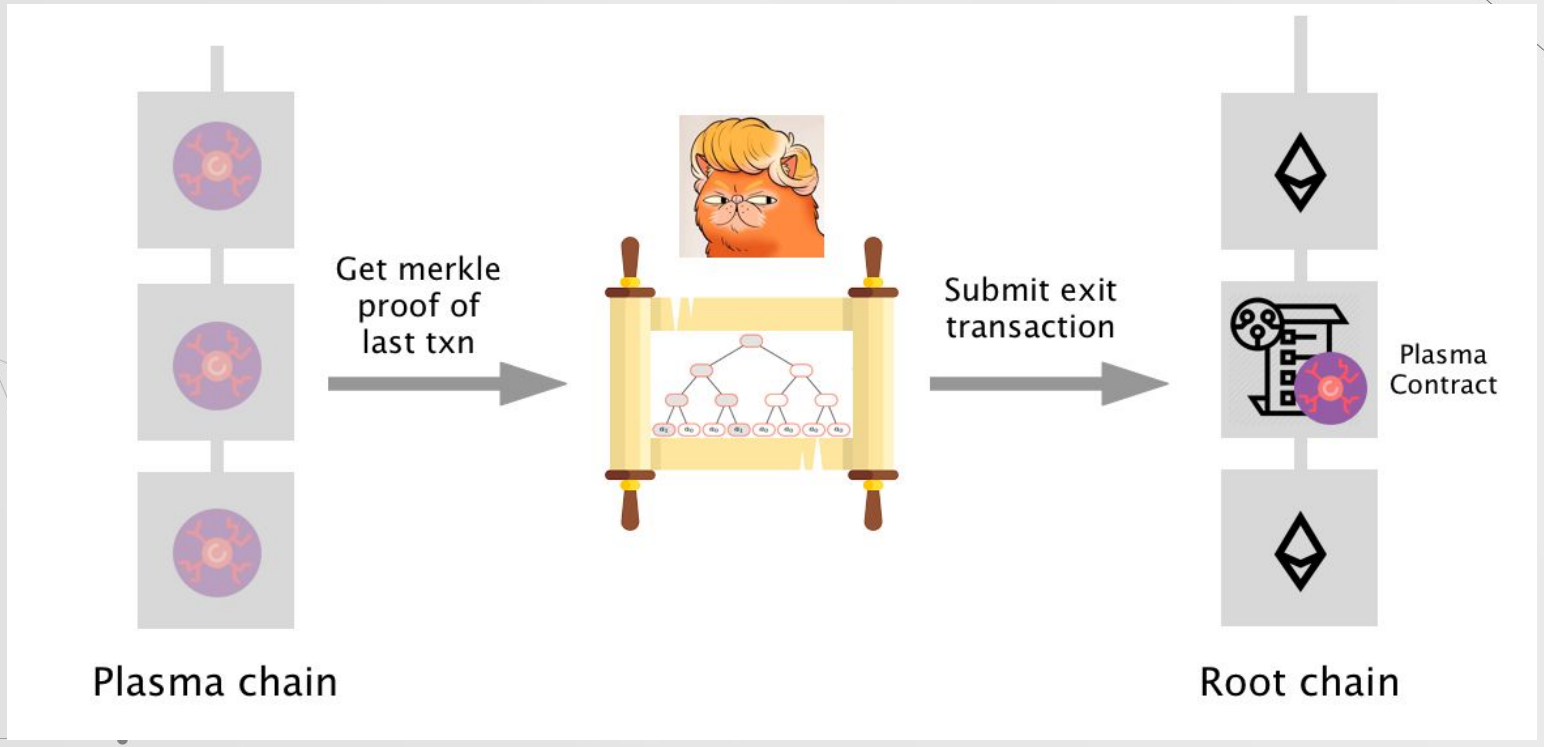
4: Kanye sends a token to Donald



- Each token you deposit is assigned a **unique ID**.
- These unique IDs are stored in a sparse **Merkle tree**.
- The index of the leaf that the coin is assigned to stores the **history of transactions** for that coin

Plasma

5: Donald could continue spending PETH or create an exit transaction to redeem for ETH on the root chain.



Plasma Implementations

- **Plasma MVP**
- **Plasma Cash**
- **Plasma Debit**
- **Plasma Prime**

https://www.learnplasma.org/en/learn/framework.html

- **Loom Network**
  Ready to go Libraries

https://loomx.io/

INTRODUCTION TO

IPFS + zk-SNARK

InterPlanetary File System

Protocol that allows to connect all computing devices in a **peer-to-peer network** with the same distributed file system. It is similar to the Web but it could be seen as a single **BitTorrent** swarm, exchanging **objects** within a **Git repository**.

IPFS
MerkleDAG

At the heart of IPFS there is the **MerkleDAG**, a directed acyclic graph whose links are hashes and sites are objects.
IPFS is essentially a P2P system for retrieving and sharing **IPFS objects.**

https://data.iota.org/#/specs

IPFS
MerkleDAG

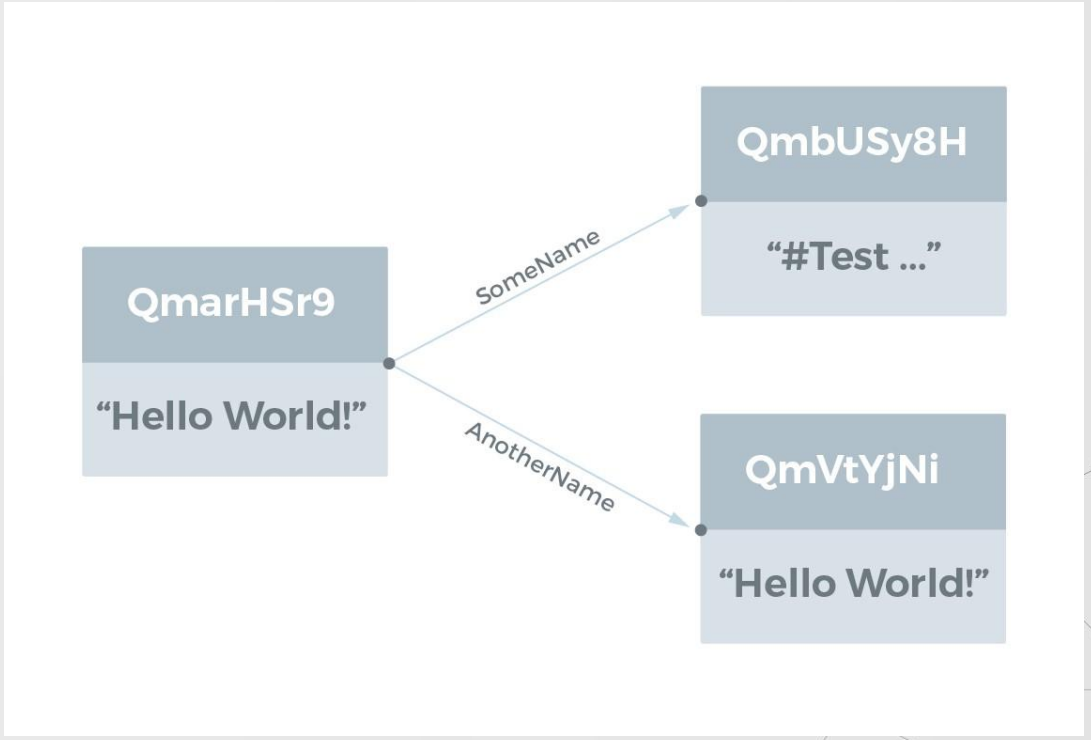
At the heart of IPFS there is the **MerkleDAG**, a directed acyclic graph whose links are hashes and sites are objects.
IPFS is essentially a P2P system for retrieving and sharing **IPFS objects.**

An IPFS object is a **data structure** with two fields:

- **Data** - unstructured binary data of size less than 256 kB
- **Links** - array of Link structures where each structure links to another IPFS object.
  A Link structure has three data fields:
    - **Name** - the name of the Link
    - **Hash** - hash of the linked IPFS object
    - **Size** - cumulative size of the linked object, including following its links

https://data.iota.org/#/specs

IPFS
Object



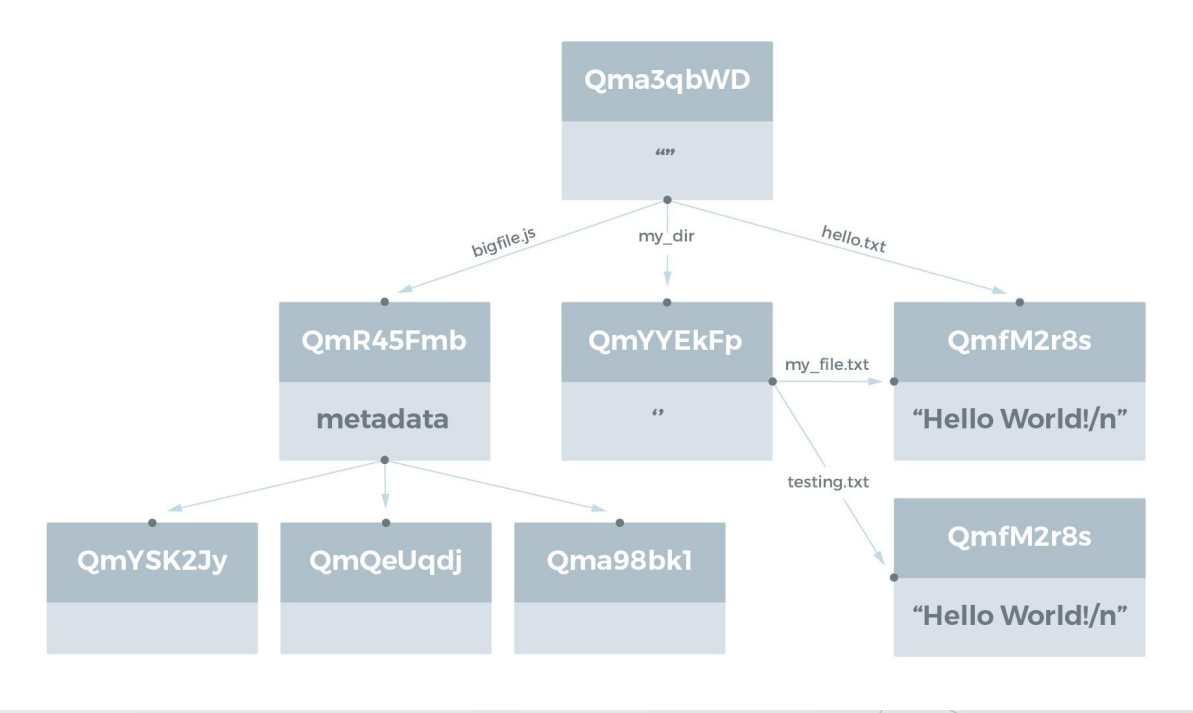
https://medium.com/@ConsenSys/an-introduction-to-ipfs-9bba4860abd0

IPFS
Files

The data and named links gives the objects collection the structure of a Merkle DAG, in which **sites are Data** and **edges are Links**. The **IPFS filesystem** is then also modelled on top of this Merkle DAG:

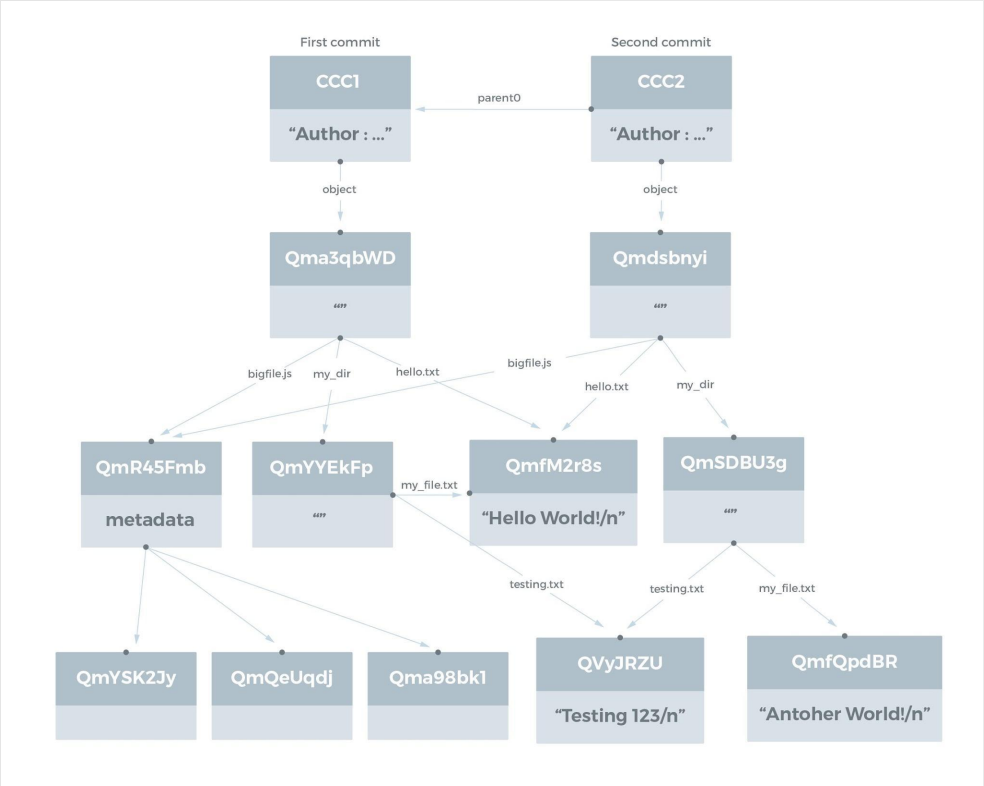
- **Small Files** (<256 kB) - represented by an IPFS object with data being the **file contents and no links**
- **Large Files** (>256 kB) - represented by a **list of links** to file blocks <256 kB
- **Directory** - represented by a list of links to IPFS objects representing files or other directories. The names of the links are the names of these files and directories

IPFS
Directory



https://medium.com/@ConsenSys/an-introduction-to-ipfs-9bba4860abd0

IPFS
Versioning



https://medium.com/@ConsenSys/an-introduction-to-ipfs-9bba4860abd0

IPFS
Block Exchange

Takes care of negotiating **bulk data transfers**, consisting in blocks that compose IPFS objects. IPFS uses **BitSwap**, a protocol inspired by BitTorrent:

- requires peers to provide the fields ***want list*** (the data a peer is looking for) and a ***have list*** (the data a peer can serve).
- acts as a persistent **marketplace**, where peers can exchange their data. Once nodes know each other and are connected this exchange protocol governs how the transfer of blocks occurs.

https://data.iota.org/#/specs

Zero Knowledge Proof

Method by which one party, **Prover**, can prove to another party, **Verifier**, that he knows a value *x* without giving any information except the fact that he knows *x*.

The Verifier goal is to prove that the Prover **possesses knowledge** of the secret parameter *x* **without revealing** the information itself or any additional information.

zk-SNARK
zero-knowledge Succinct Non-interactive ARguments of Knowledge

Consists of a zero-knowledge proof where **no interaction** is necessary between Prover and Verifier.

This constitutes the backbone of the **Zcash** protocol, one of the most used cryptocurrency that provide enhanced **privacy** for its users compared to Ethereum and Bitcoin

zk-SNARK Definition

A zk-SNARK consists of three algorithms G, P, V defined as follows:

zk-SNARK Definition

A zk-SNARK consists of three algorithms G, P, V defined as follows:

Program A program C:

$$C(x, w) \longrightarrow \{0, 1\}$$

takes as input a public value x and a secret witness w and returns a boolean value.

zk-SNARK Definition

A zk-SNARK consists of three algorithms G, P, V defined as follows:

Program A program C:

$$C(x, w) \longrightarrow \{0, 1\}$$

takes as input a public value x and a secret witness w and returns a boolean value.

Generator A Key Generator G:

$$(pk, vk) = G(C, \lambda)$$

takes as input a program C and a secret parameter λ and generates two public keys, a proving key pk and a validation key vk. These keys are public parameters that only need to be generated once for a given C. The λ parameter is called "toxic waste" because anyone who knows this parameter can generate fake proofs, hence must be destroyed during the process.

zk-SNARK
Definition

Prover A Prover algorithm P:

$$proof = P(pk, x, w)$$

takes as input the proving key pk, the public input x and the secret witness w and generates a *proof* that the Prover knows a witness w and that the witness satisfies the program.

zk-SNARK
Definition

Prover A Prover algorithm P:

$$proof = P(pk, x, w)$$

takes as input the proving key pk, the public input x and the secret witness w and generates a $proof$ that the Prover knows a witness w and that the witness satisfies the program.

Verifier A Verifier algorithm V:

$$V(vk, x, proof) \longrightarrow \{0, 1\}$$

takes as input the verification key vk, the public input x and the $proof$ and returns true if the Prover knows a witness w satisfying $C(x, w) == 1$, false otherwise.

RESOURCES

1. IOTA Applications https://docs.iota.org/
   - Masked Authenticated Messaging https://github.com/iotaledger/mam.client.js
   - Data Marketplace https://data.iota.org
2. State Channels
   - Ethereum µRaiden https://raiden.network/
   - Flash Channels https://blog.iota.org/instant-feeless-flash-channels-88572d9a4385
   - Ethereum Plasma https://plasma.io/plasma.pdf
3. Introduction to
   - IPFS https://ipfs.io/
   - zk-SNARK https://z.cash/technology/zksnarks/