IOTA WORKSHOP

IOTA Advanced Concepts

Speaker(s): Sreenivas Chinni & Dharmen Dulla

Event Organizers

dcentrum Community

Connect | Collaborate | Create

Venue Support

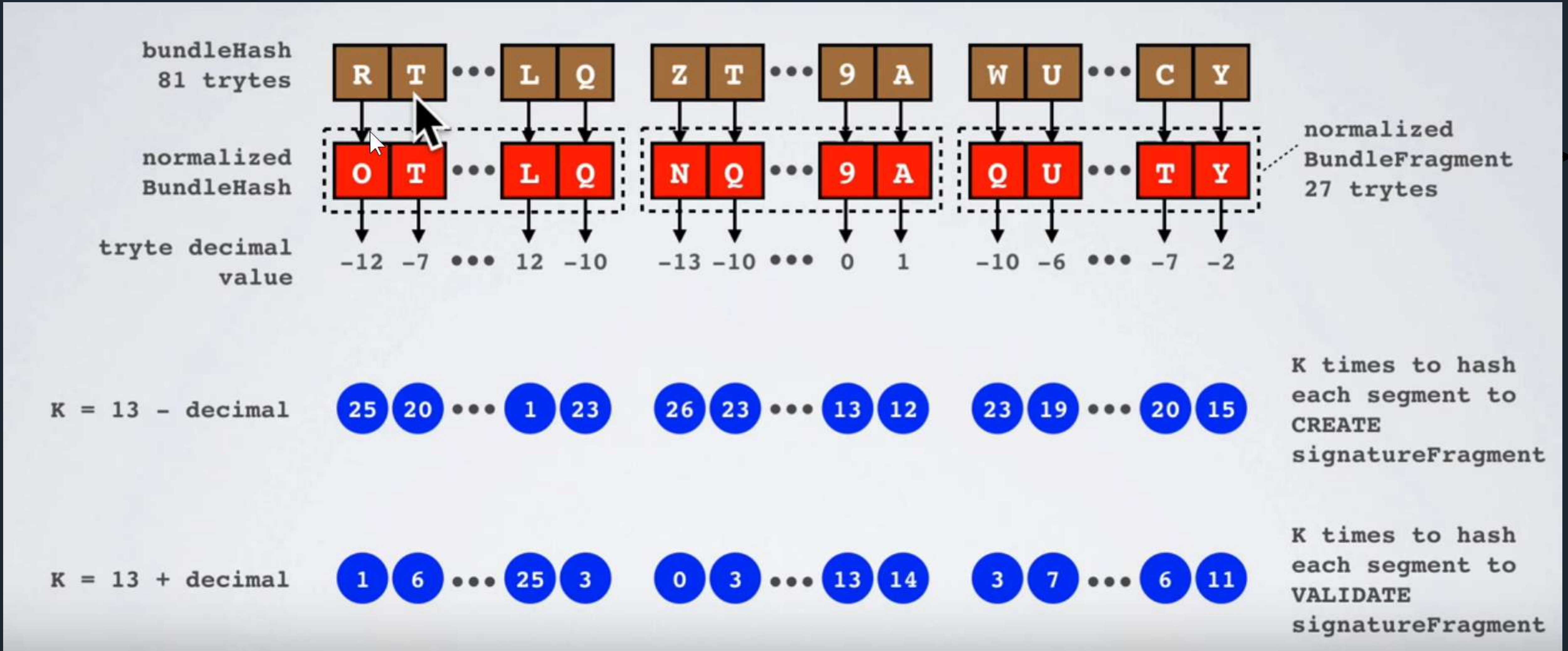
Tech Mahindra

Agenda

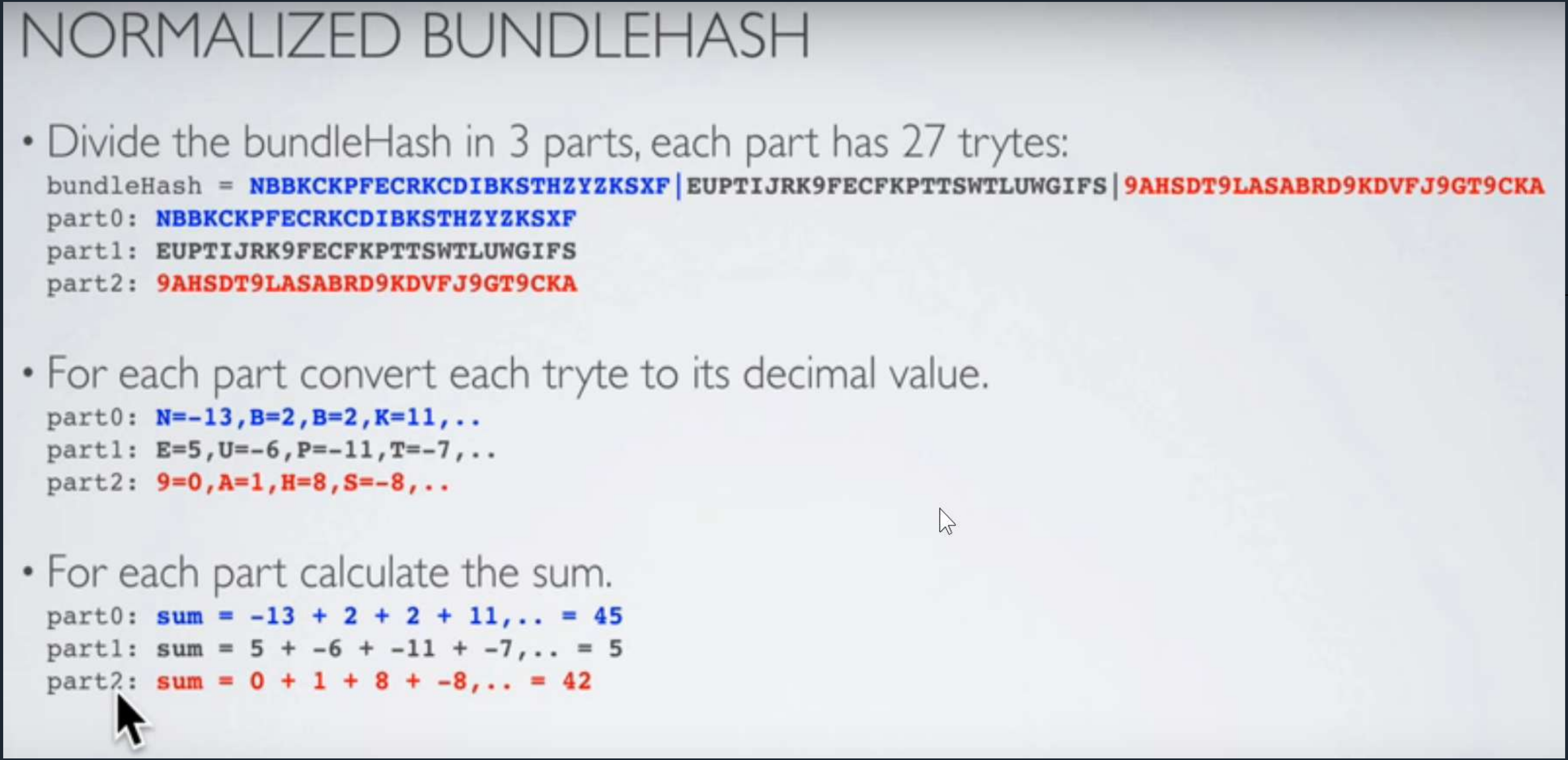
- ❑Quantum Resilient
- ❑Transaction Flow
- ❑MAM Channels
- ❑IOT device interaction with IOTA

Quantum Resilient

- IOTA uses the Hash based one time signatures
- How does Winternitz Signature work?



Quantum Resilient



NORMALIZED BUNDLEHASH

- Divide the bundleHash in 3 parts, each part has 27 trytes:

```
bundleHash = NBBKCKPFECRKCDIBKSTHZYZKSXF|EUPTIJRK9FECFKPTTSWTLUWGIFS|9AHSDT9LASABRD9KDVFJ9GT9CKA
part0: NBBKCKPFECRKCDIBKSTHZYZKSXF
part1: EUPTIJRK9FECFKPTTSWTLUWGIFS
part2: 9AHSDT9LASABRD9KDVFJ9GT9CKA
```

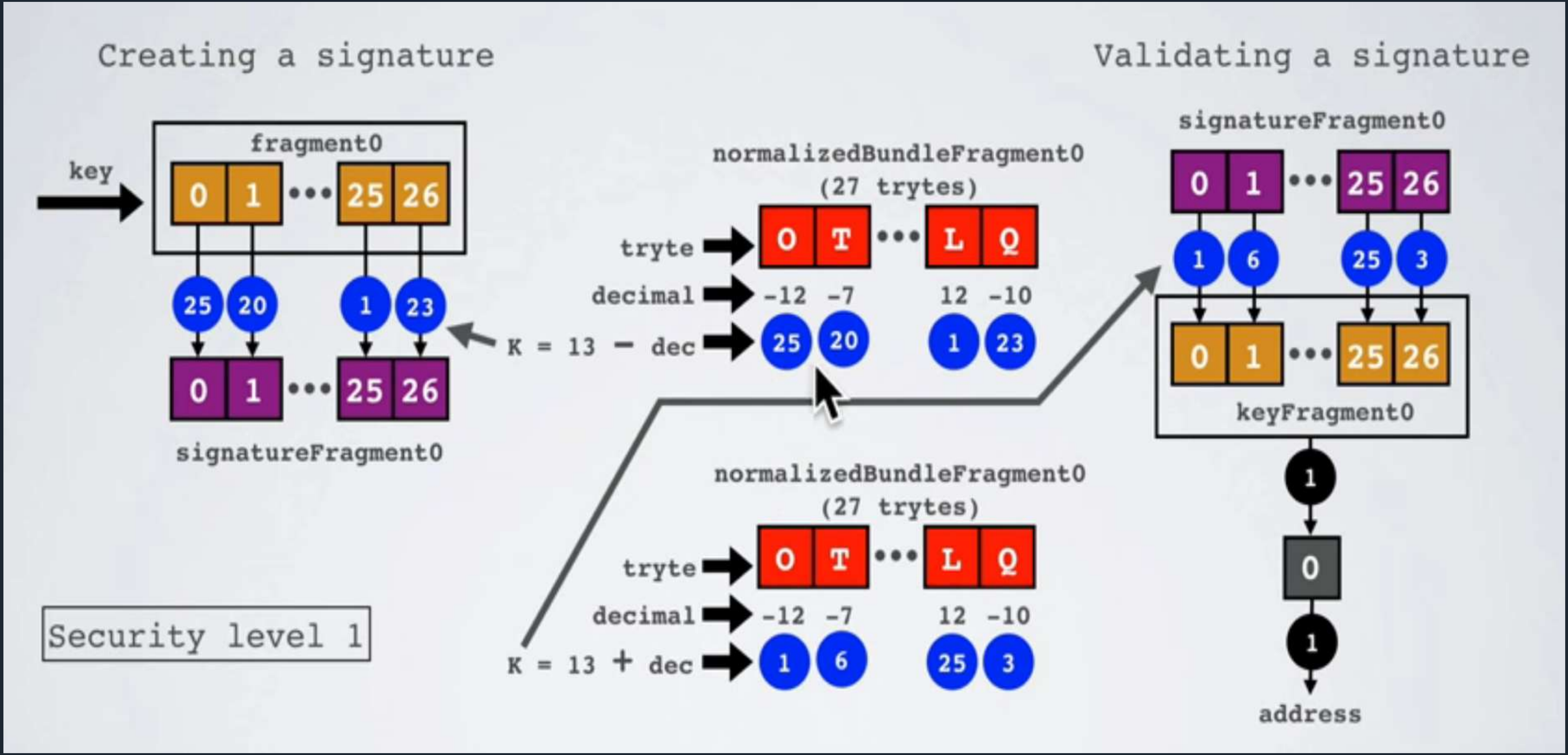
- For each part convert each tryte to its decimal value.

```
part0: N=-13,B=2,B=2,K=11,..
part1: E=5,U=-6,P=-11,T=-7,..
part2: 9=0,A=1,H=8,S=-8,..
```

- For each part calculate the sum.

```
part0: sum = -13 + 2 + 2 + 11,.. = 45
part1: sum = 5 + -6 + -11 + -7,.. = 5
part2: sum = 0 + 1 + 8 + -8,.. = 42
```

Quantum Resilient



IOTA Transaction Bundles

A Transaction is a record with multiple attributes like address, value etc. It may be for depositing IOTAs or withdrawal of IOTAs or sending message

Transaction Attributes (2673 trytes)

- ❑ signatureMessageFragment
- ❑ Address
- ❑ value
- ❑ timestamp
- ❑ currentIndex
- ❑ lastIndex
- ❑ bundle
- ❑ trunkTransaction
- ❑ branchTransaction
- ❑ Nonce

Transaction Types

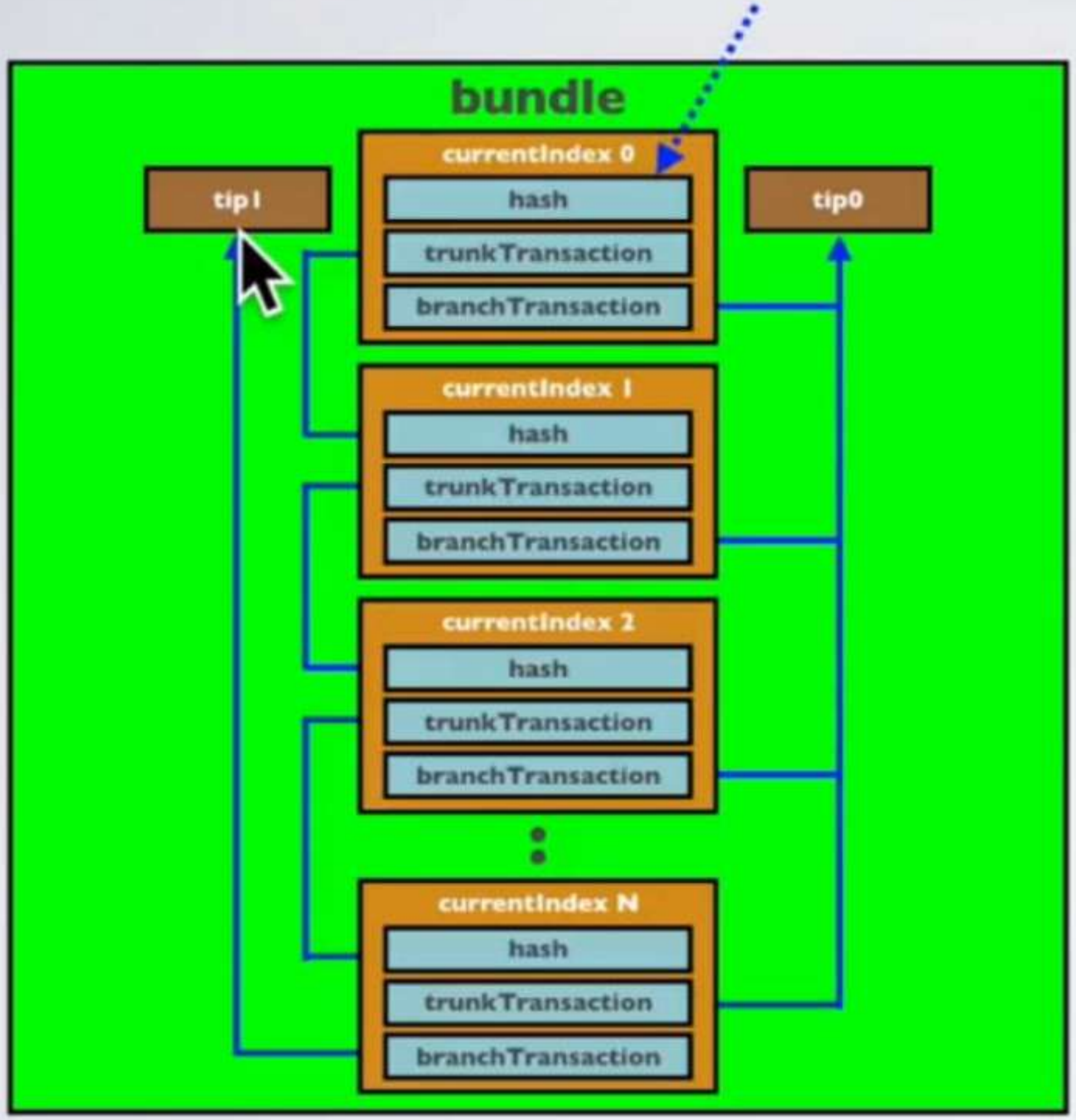
- ❑ Input transaction (withdraw)
- ❑ Output transaction (Deposit / message)
- ❑ Meta Transaction (for signature fragments)

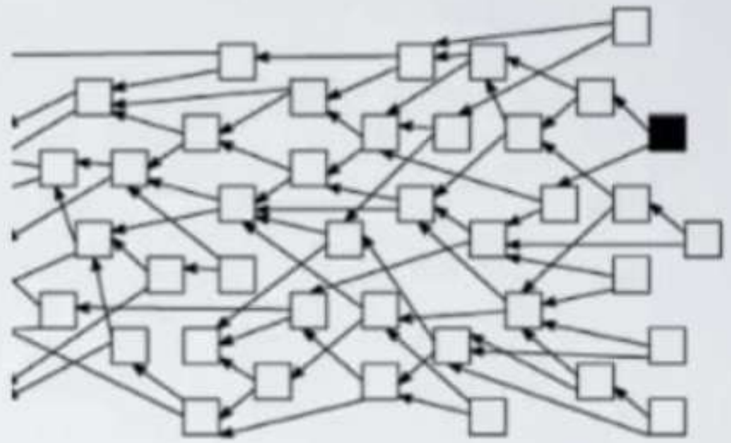
Transaction Bundle structure

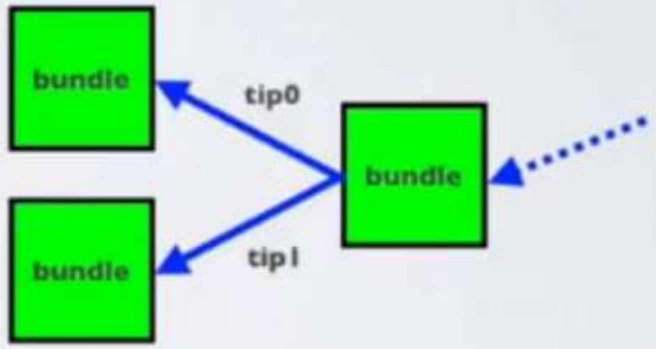
A bundle is an atomic unit of one or more transactions

index	Contents	Transaction type
0	Recipient's address, positive value and sign	output / Deposit
1	Sender's address and the first part of its signature Negative value	Input / withdrawal
2	Sender's address and the second part of its signature Negative or zero value	Input / withdrawal
3	Sender's address and the rest of its signature	Meta transaction
4	Sender's address and positive value	Output / deposit

TRANSACTIONS IN BUNDLE



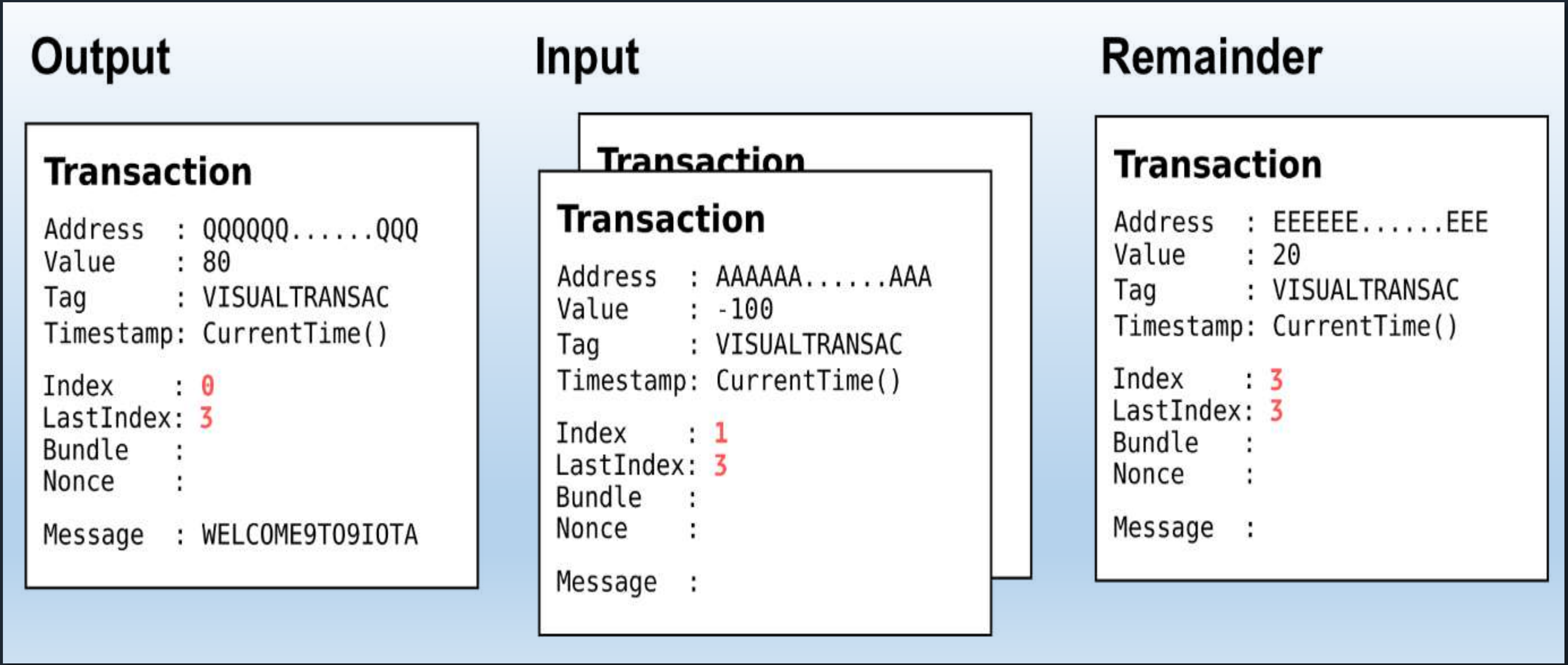




```
currentIndex 0 = tail transaction
currentIndex N = head transaction
```

Prepare transactions

Prepare one or more transaction with senders or receivers addresses and values sign all input transaction with sender's keys using key generator



Output

```
Transaction

Address  : QQQQQQ......QQQ
Value    : 80
Tag      : VISUALTRANSAC
Timestamp: CurrentTime()

Index    : 0
LastIndex: 3
Bundle   :
Nonce    :

Message  : WELCOME9TO9IOTA
```

Input

```
Transaction

Transaction

Address  : AAAAAA......AAA
Value    : -100
Tag      : VISUALTRANSAC
Timestamp: CurrentTime()

Index    : 1
LastIndex: 3
Bundle   :
Nonce    :

Message  :
```

Remainder

```
Transaction

Address  : EEEEEE......EEE
Value    : 20
Tag      : VISUALTRANSAC
Timestamp: CurrentTime()

Index    : 3
LastIndex: 3
Bundle   :
Nonce    :

Message  :
```

Sign Transactions

Sign all input transactions with private key

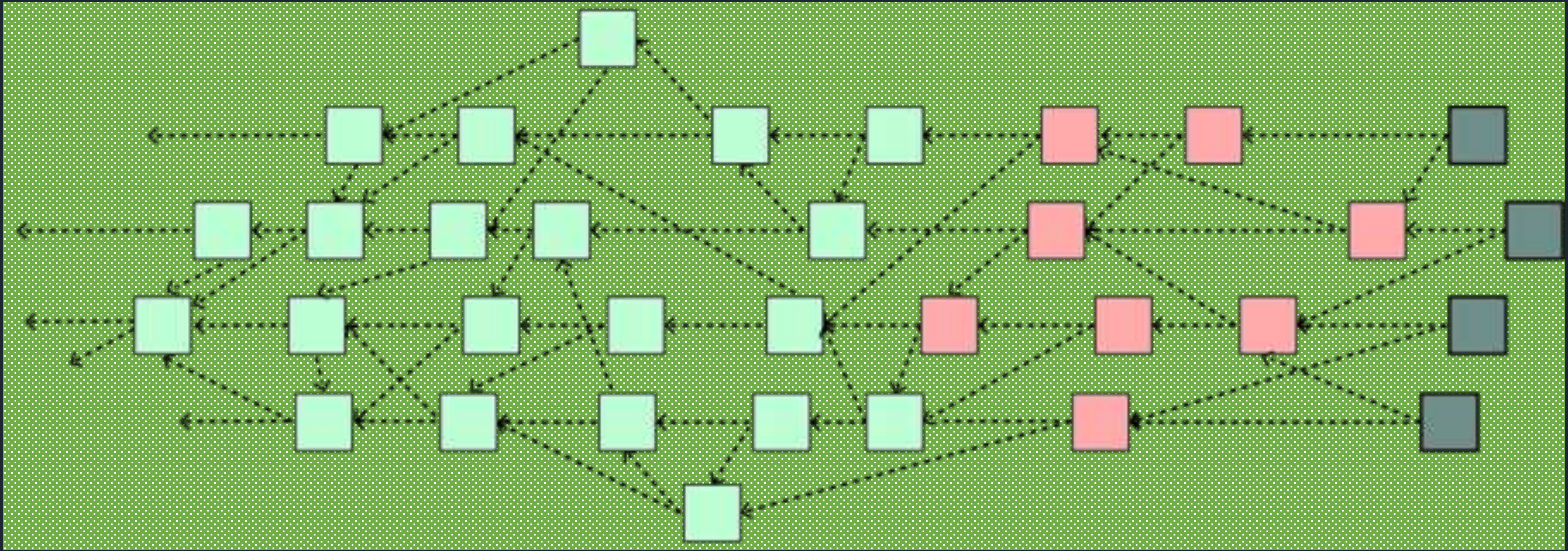
- Seed 81 trytes
- Private and Public Key (Address)
- Signature for input transactions
  - Hash based One time signature (Winternitz) using Signature Fragment Generator
  - Use Address only once to send IOTAs

Prepare bundle

1. Generate bundle hash.
2. Assign bundle hash to transactions

Tip Selection

1. Every transaction bundle should reference to two previous unconfirmed transaction as parent transaction
2. Process used MCMC algorithm to pickup these tip selections



Proof of work

1. Get trytes of transaction
2. Get minimum weight magnitude (this is 9 for test net and 14 for mainnet)
3. Execute pow function with MWM and transaction trytes
4. The above step returns the nonce.
5. Include the nonce with transaction trytes
6. Execute curl hash with the new transaction trytes. This returns the transaction hash

7. If the this hash matching difficulty no of 9s at the end, then transaction is valid, otherwise repeat the same process by incrementing the nonce

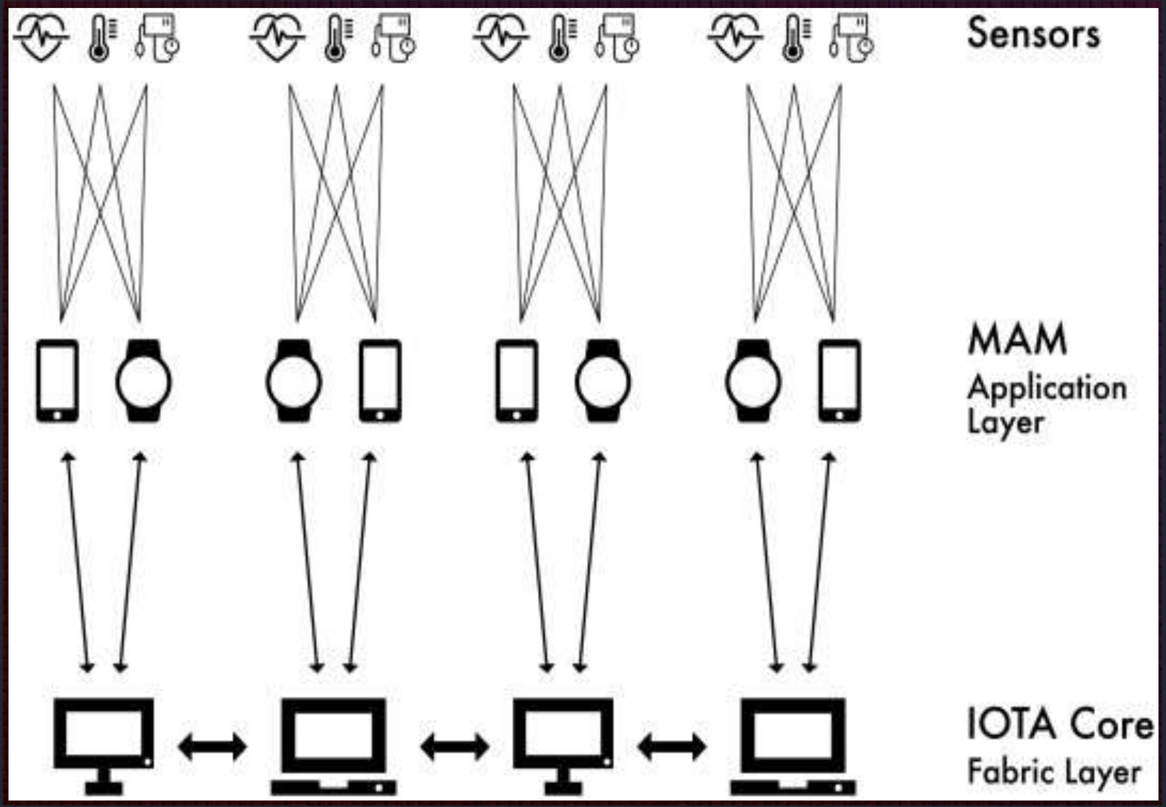
```
pow( ttrytes, mwm ) = nonce,
trans_hash = ‘’
white trans_hash.endswith(‘9999’) == false
        trans_hash = curlhash( ttrytes+nonce +1)
```

1. Broadcast the bundle to the tangle

Masked Authenticated Message

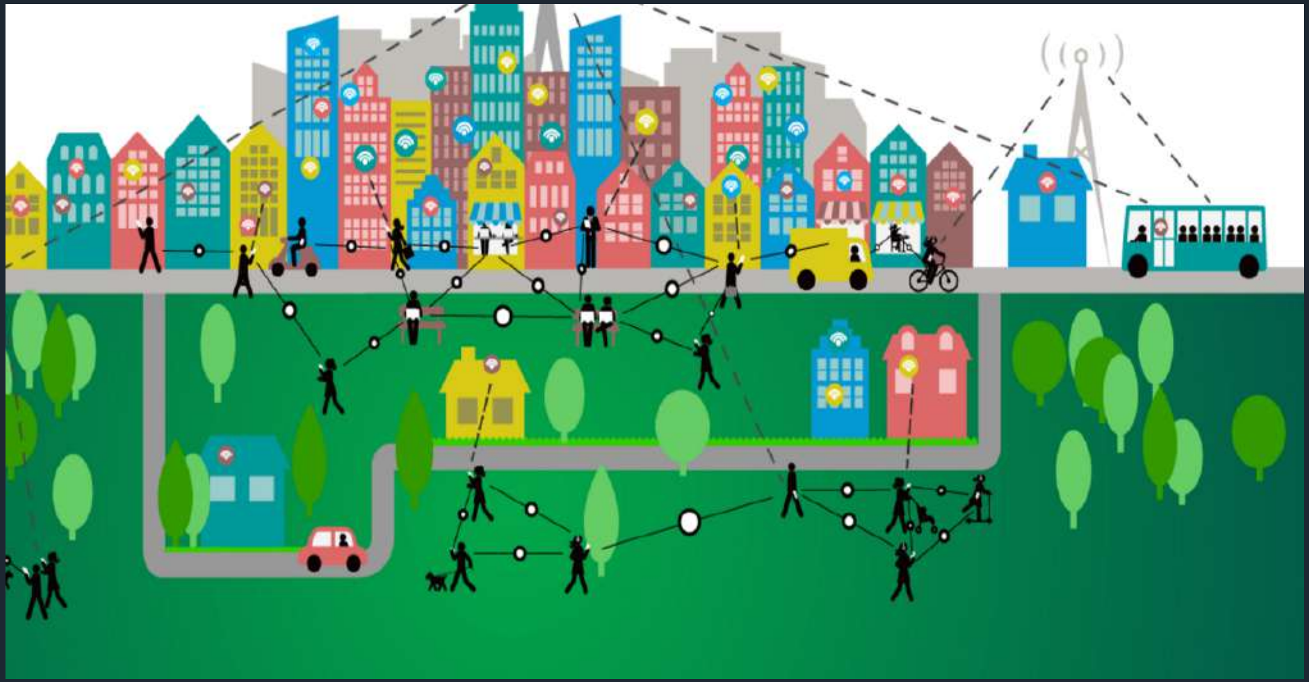
- ❑ Masked= message is encrypted
- ❑ Authenticated= message is confirmed to be coming from the device
- ❑ Messaging = stream of the data is created on the tangle

"Masked Authenticated Messaging is used to send fully encrypted message from Authenticated parties"



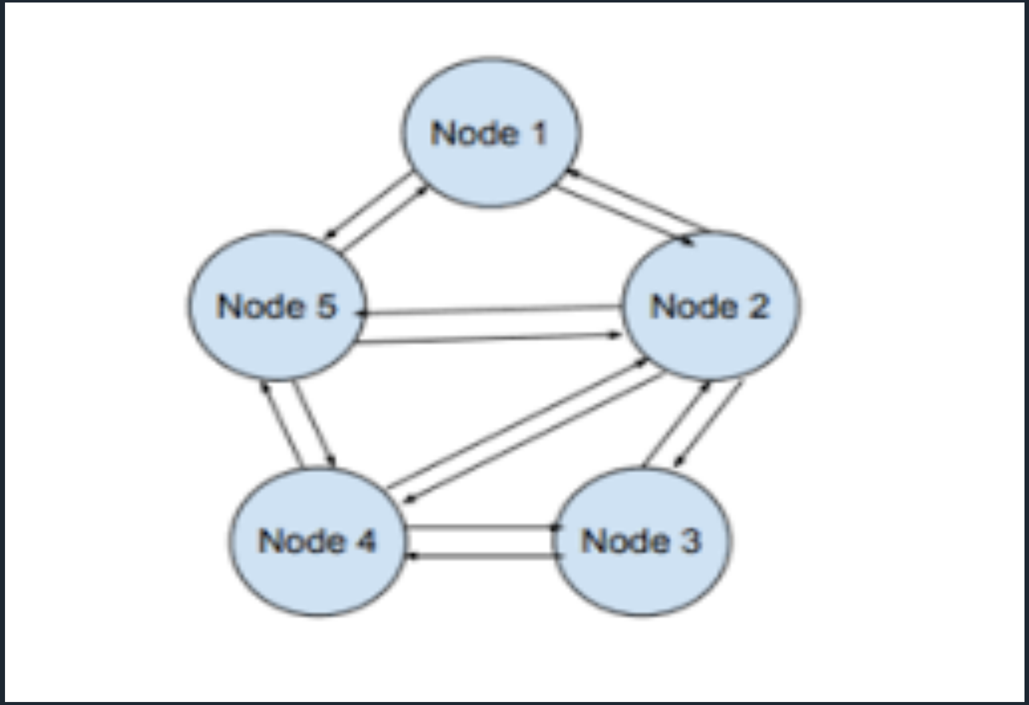
Use cases

- ❑ Encrypted message streaming
- ❑ Machine to Machine communication
 (cars in traffic)
- ❑ Pub-Sub kind of data marketplace
- ❑ An enterprise can instruct their IoT
 devices for firmware or configuration
updates.
poll their existence like charging station
poll it’s location for cars



Gossip protocol

- Iota uses the gossip protocol to propagate messages through the network.
- Any new transaction is immediately broadcast to all neighbors.
- When a node detect that a transaction is missing (i.e. the branch or trunk of a new transaction): the node will ask for the missing tx to it's neighbors.

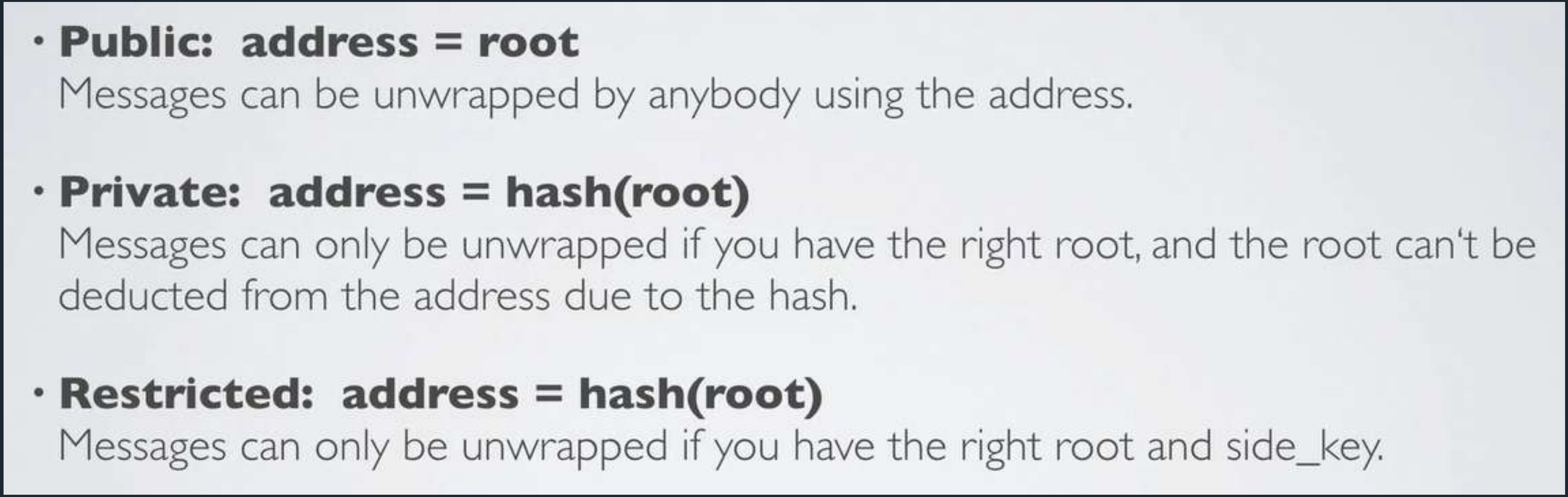


Security & Privacy - Channels

Channels are the named message streams that targets particular kind of audience like any other streaming platforms have. Ex: YouTube, slack or discord.

channel = {side_key: null, mode: 'public',next_root: null,security,start: 0,count: 1,next_count: 1,index: 0}

Every channel has mode of visibility that ensures the privacy of the message.



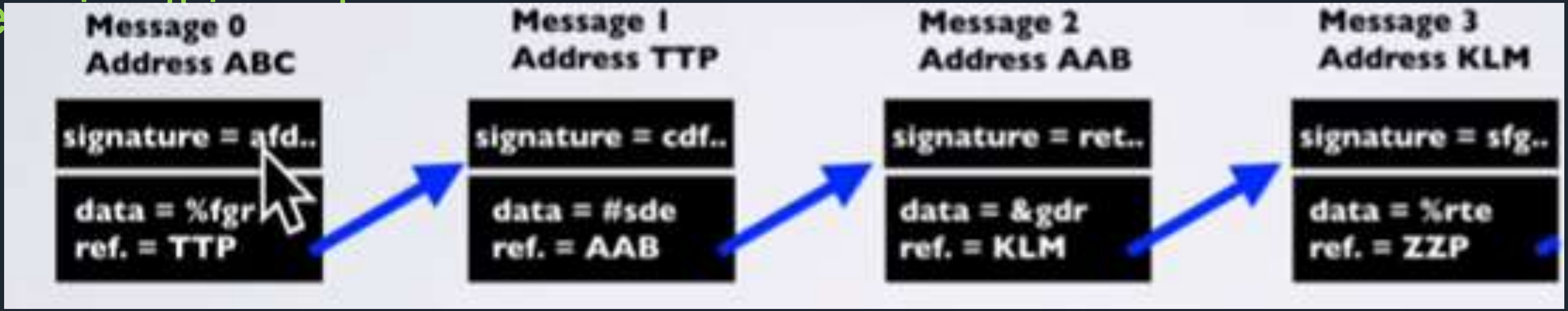
- **Public: address = root**
  Messages can be unwrapped by anybody using the address.

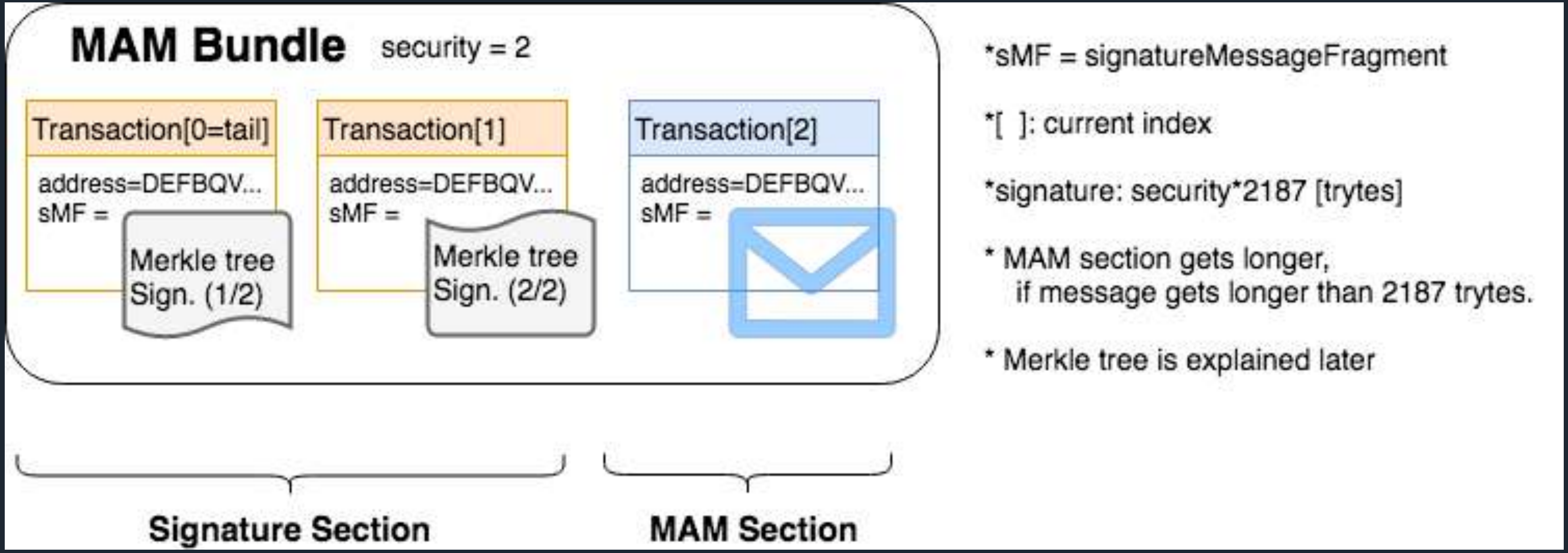
- **Private: address = hash(root)**
  Messages can only be unwrapped if you have the right root, and the root can't be deducted from the address due to the hash.

- **Restricted: address = hash(root)**
  Messages can only be unwrapped if you have the right root and side_key.

Message Chain

On message stream every message holds a reference to the next message. the message stream only flows one direction .In the stream the encrypted message and also contains a signature. channel id is also called the root the message is attached to the tangle





*sMF = signatureMessageFragment

*[ ]: current index

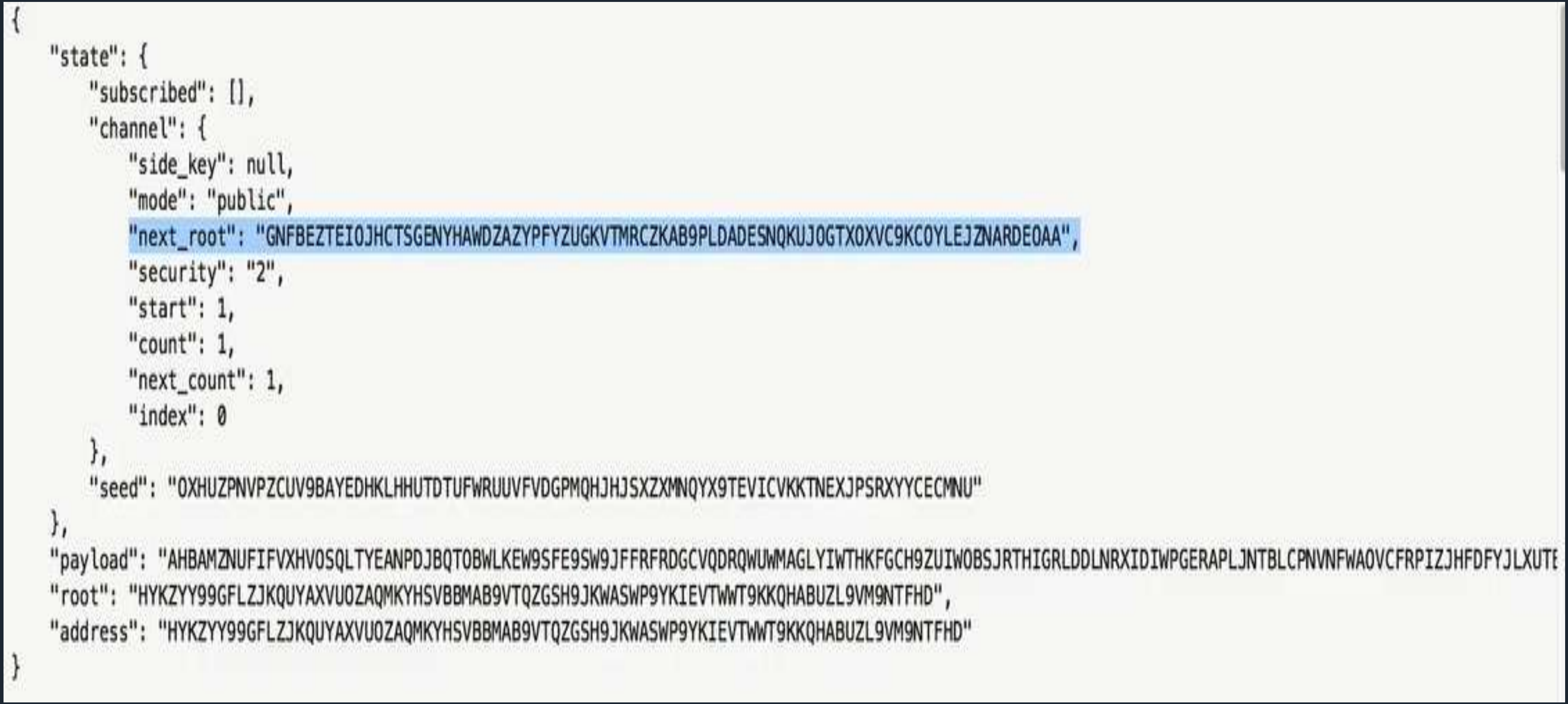
*signature: security*2187 [trytes]

* MAM section gets longer, if message gets longer than 2187 trytes.

* Merkle tree is explained later

Public Mode

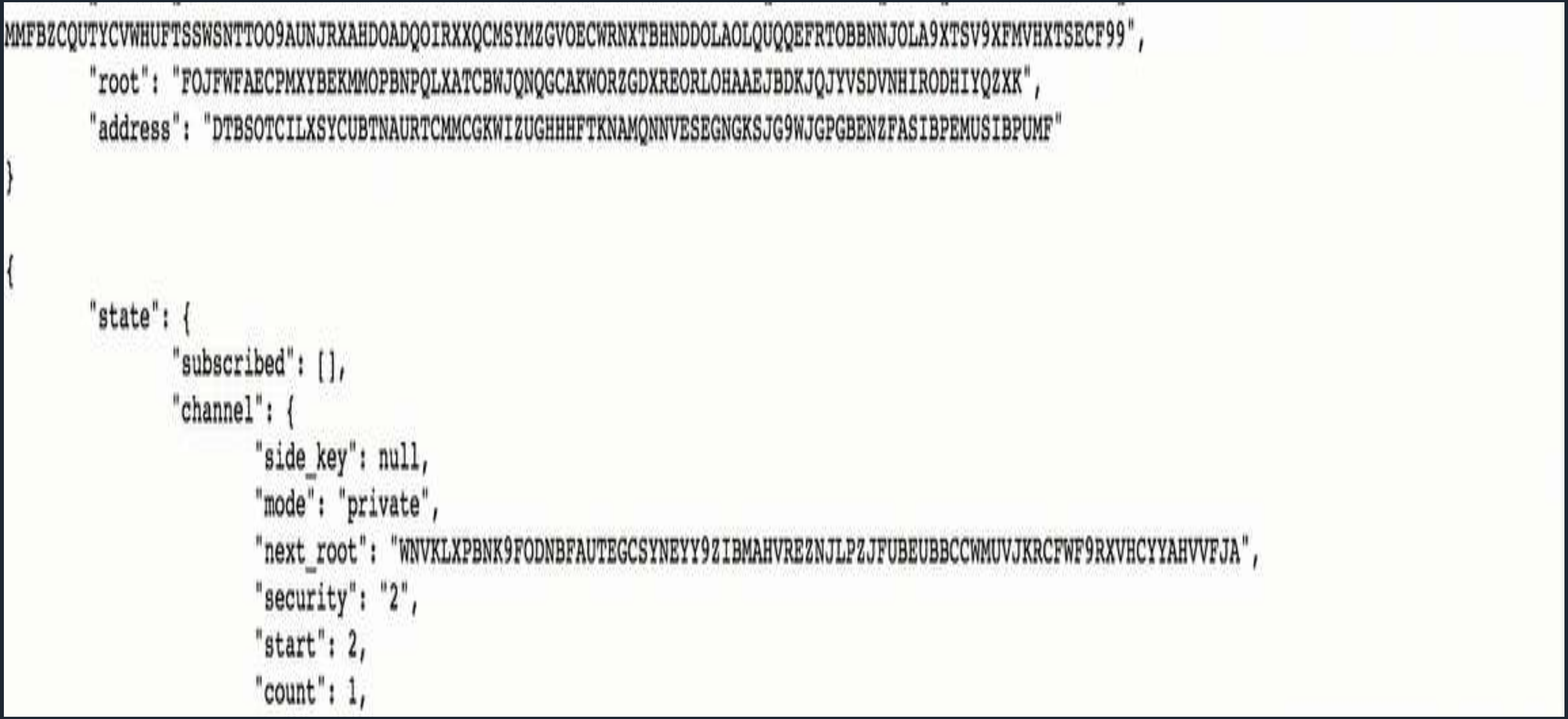
Everyone can view.



```
{
    "state": {
        "subscribed": [],
        "channel": {
            "side_key": null,
            "mode": "public",
            "next_root": "GNFBEZTEIOJHCTSGENYHAWDZAZYPFYZUGKVTMRCZKAB9PLDADESNQKUJOGTXOXVC9KCOYLEJZNARDEOAA",
            "security": "2",
            "start": 1,
            "count": 1,
            "next_count": 1,
            "index": 0
        },
        "seed": "OXHUZPNVPZCUV9BAYEDHKLHHUTDTUFWRUUVFVDGPMQHJHJSXZXMNQYX9TEVICVKKTNEXJPSRXYYCECMNU"
    },
    "payload": "AHBAMZNUFIFVXHVOSQLTYEANPDJBQTOBWLKEW9SFE9SW9JFFRFRDGCVQDRQWUWMAGLYIWTHKFGCH9ZUIWOBSJRTHIGRLDDLNRXIDIWPGERAPLJNTBLCPNVNFWAOVCFRPIZJHFDFYJLXUTE
    "root": "HYKZYY99GFLZJKQUYAXVUOZAQMKYHSVBBMAB9VTQZGSH9JKWASWP9YKIEVTWWT9KKQHABUZL9VM9NTFHD",
    "address": "HYKZYY99GFLZJKQUYAXVUOZAQMKYHSVBBMAB9VTQZGSH9JKWASWP9YKIEVTWWT9KKQHABUZL9VM9NTFHD"
}
```

Private Mode

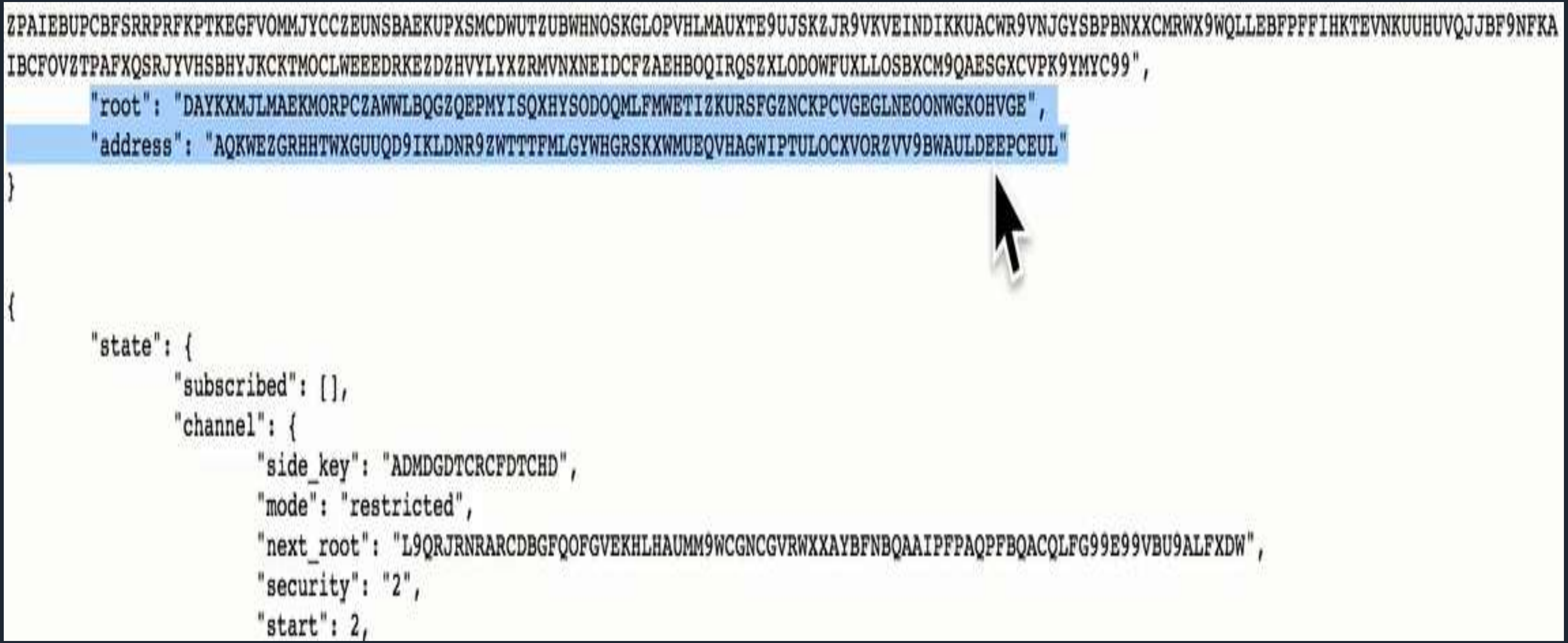
Only you (i.e. seed owner) can view.



```
MMFBZCQUTYCVWHUFTSSWSNTTOO9AUNJRXAHDOADQOIRXXQCMSYMZGVOECWRNXTBHNDDOLAOLQUQQEFRTOBBNNJOLA9XTSV9XFMVHXTSECF99",
        "root": "FOJFWFAECPMXYBEKMMOPBNPQLXATCBWJQNQGCAKWORZGDXREORLOHAAEJBDKJQJYVSDVNHIRODHIYQZXK",
        "address": "DTBSOTCILXSYCUBTNAURTCMMCGKWIZUGHHHFTKNAMQNNVESEGNGKSJG9WJGPGBENZFASIBPEMUSIBPUMF"
}


{
        "state": {
                "subscribed": [],
                "channel": {
                        "side_key": null,
                        "mode": "private",
                        "next_root": "WNVKLXPBNK9FODNBFAUTEGCSYNEYY9ZIBMAHVREZNJLPZJFUBEUBBCCWMUVJKRCFWF9RXVHCYYAHVVFJA",
                        "security": "2",
                        "start": 2,
                        "count": 1,
```

Restricted Mode

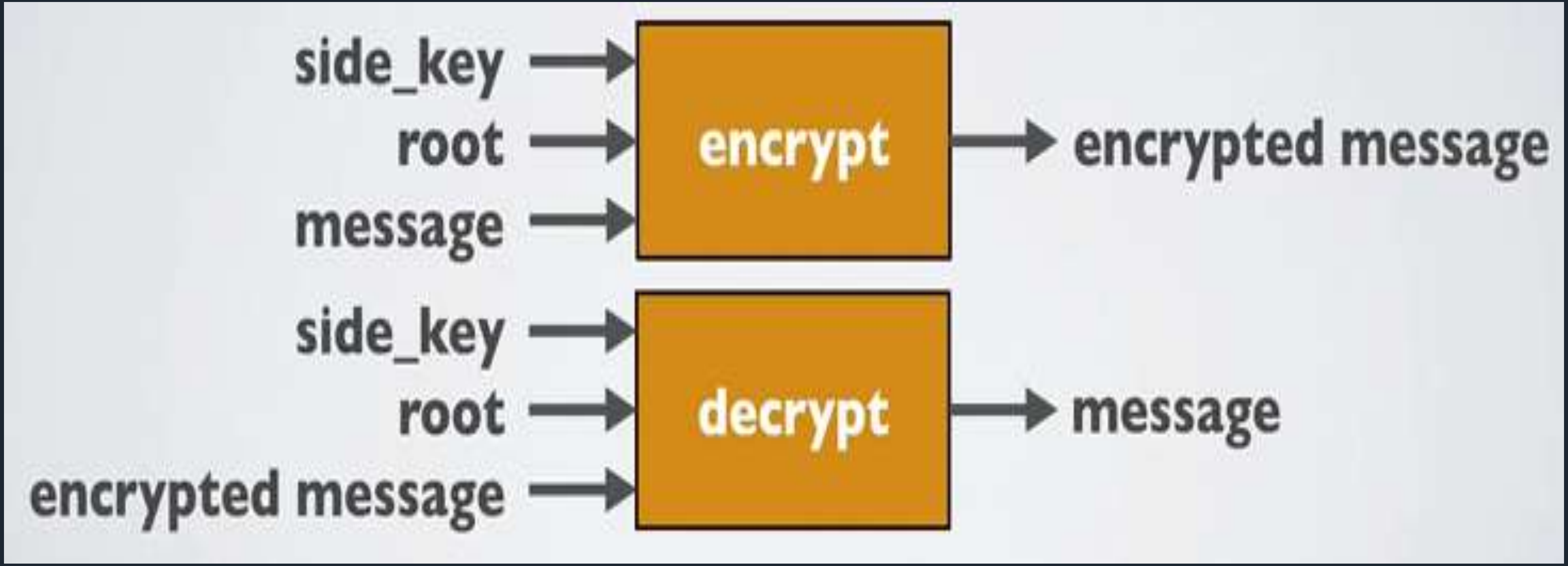
Only you can specify your viewers by telling them a key. This key is named as sideKey in source code



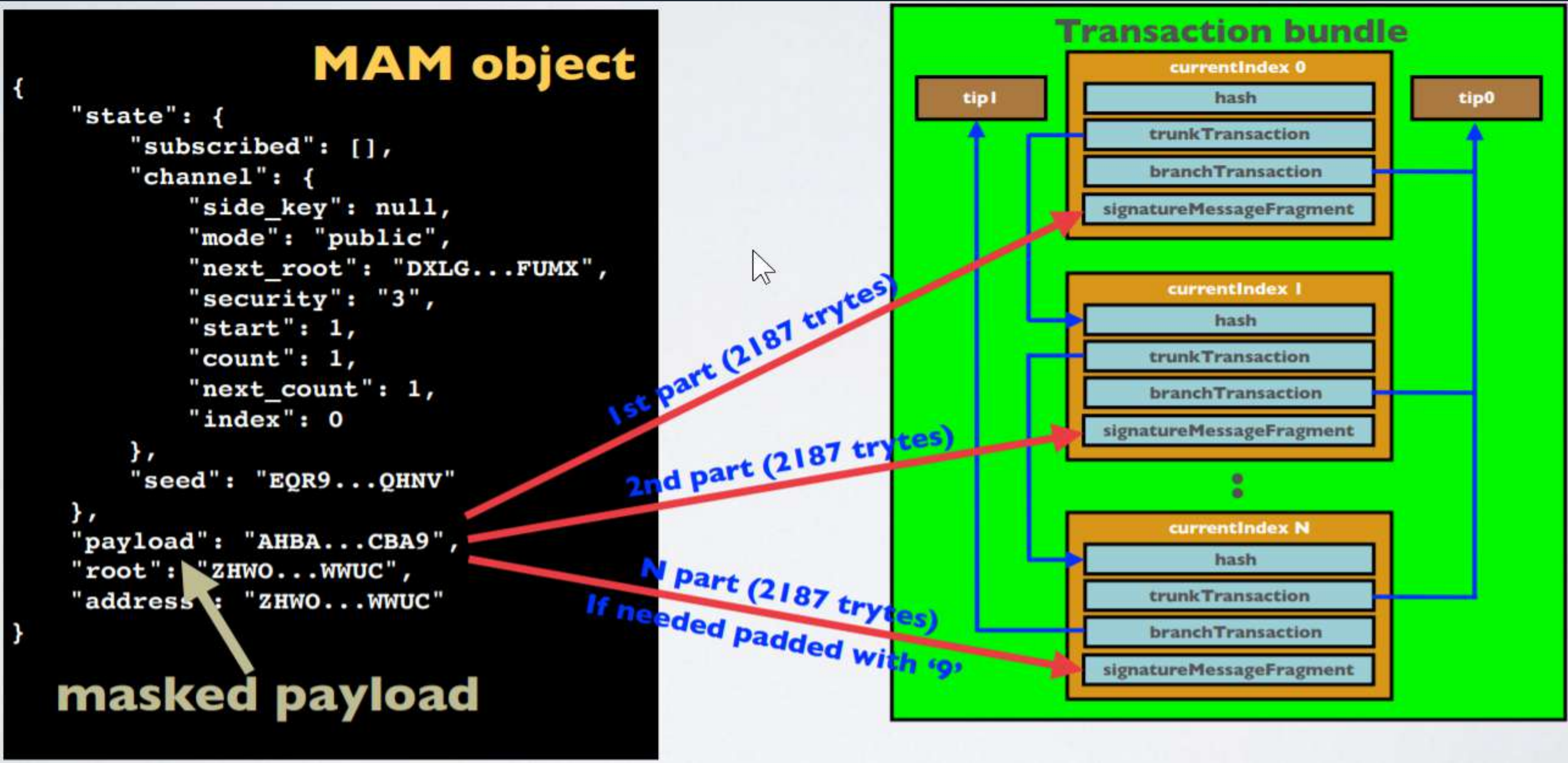
```
ZPAIEBUPCBFSRRPRFKPTKEGFVOMMJYCCZEUNSBAEKUPXSMCDWUTZUBWHNOSKGLOPVHLMAUXTE9UJSKZJR9VKVEINDIKKUACWR9VNJGYSBPBNXXCMRWX9WQLLEBFPFFIHKTEVNKUUHUVQJJBF9NFKA
IBCFOVZTPAFXQSRJYVHSBHYJKCKTMOCLWEEEDRKEZDZHVYLYXZRMVNXNEIDCFZAEHBOQIRQSZXLODOWFUXLLOSBXCM9QAESGXCVPK9YMYC99",
        "root": "DAYKXMJLMAEKMORPCZAWWLBQGZQEPMYISQXHYSODOQMLFMWETIZKURSFGZNCKPCVGEGLNEOONWGKOHVGE",
        "address": "AQKWEZGRHHTWXGUUQD9IKLDNR9ZWTTTFMLGYWHGRSKXWMUEQVHAGWIPTULOCXVORZVV9BWAULDEEPCEUL"
}


{
        "state": {
                "subscribed": [],
                "channel": {
                        "side_key": "ADMDGDTCRCFDTCHD",
                        "mode": "restricted",
                        "next_root": "L9QRJRNRARCDBGFQOFGVEKHLHAUMM9WCGNCGVRWXXAYBFNBQAAIPFPAQPFBQACQLFG99E99VBU9ALFXDW",
                        "security": "2",
                        "start": 2,
```

Side Key

The side is used to encrypt and decrypt the message and The side_key is required when using the restricted mode



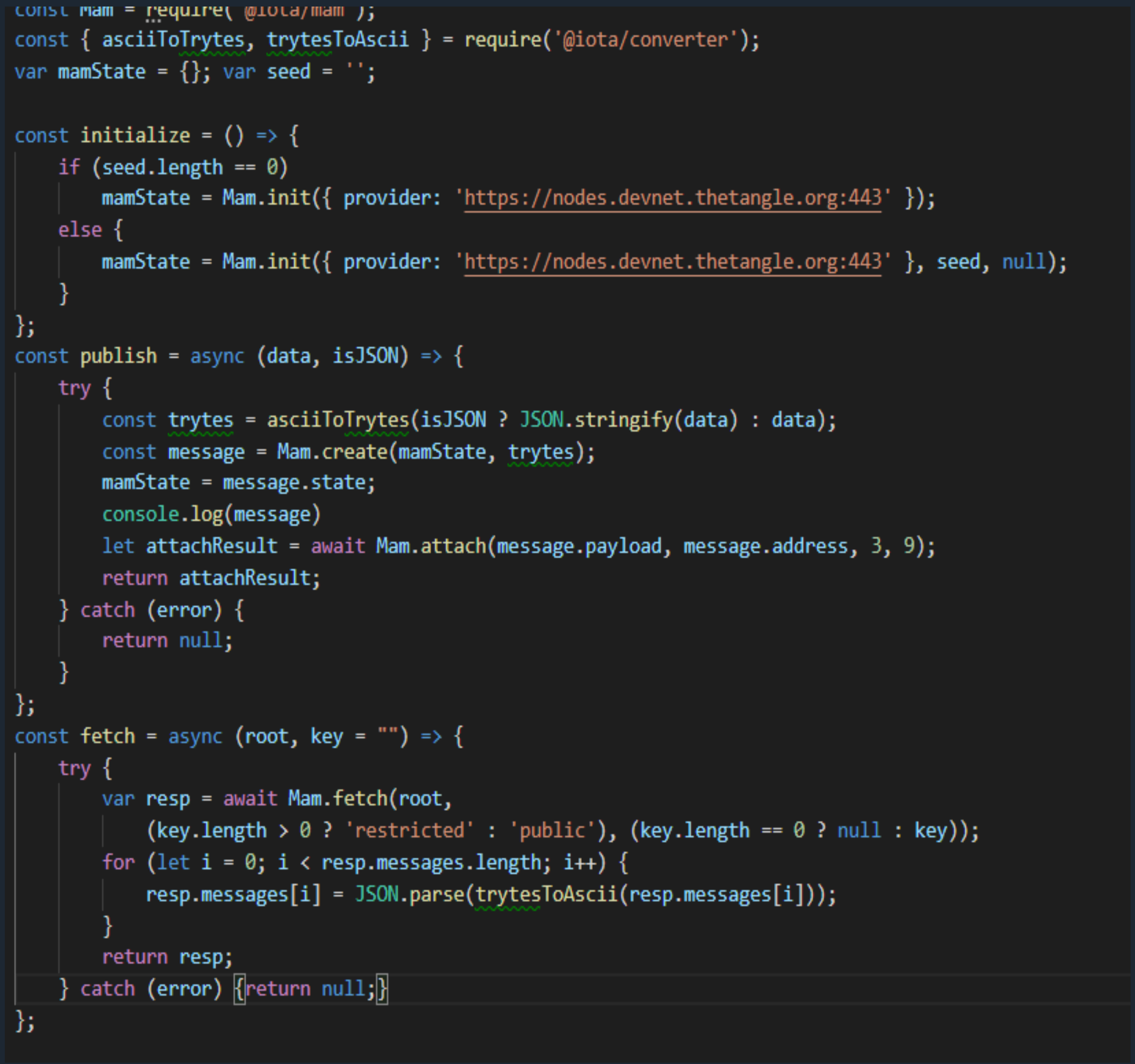
MAM



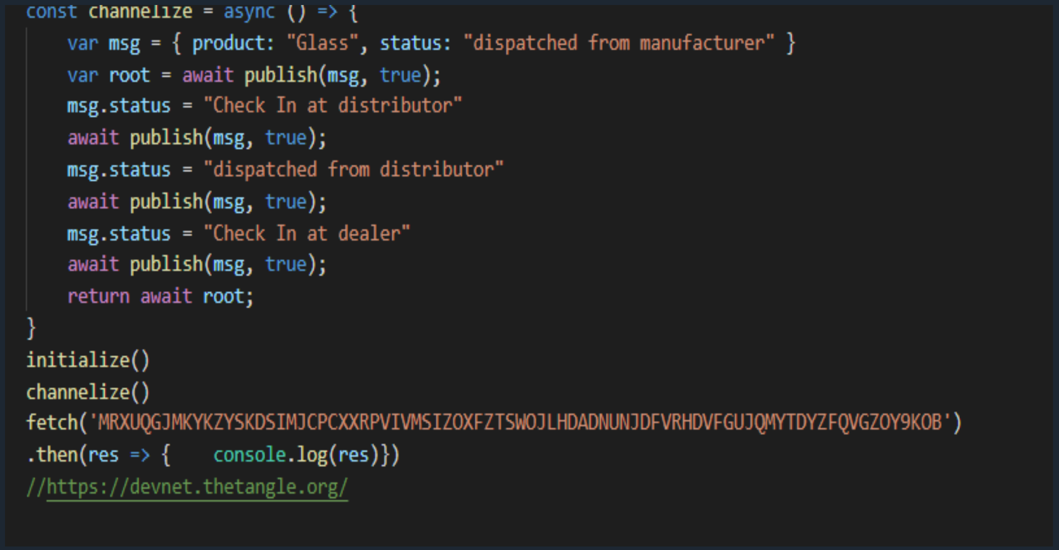
MAM object

```
{
    "state": {
        "subscribed": [],
        "channel": {
            "side_key": null,
            "mode": "public",
            "next_root": "DXLG...FUMX",
            "security": "3",
            "start": 1,
            "count": 1,
            "next_count": 1,
            "index": 0
        },
        "seed": "EQR9...QHNV"
    },
    "payload": "AHBA...CBA9",
    "root": "ZHWO...WWUC",
    "address": "ZHWO...WWUC"
}
```

MAM – Sample Code

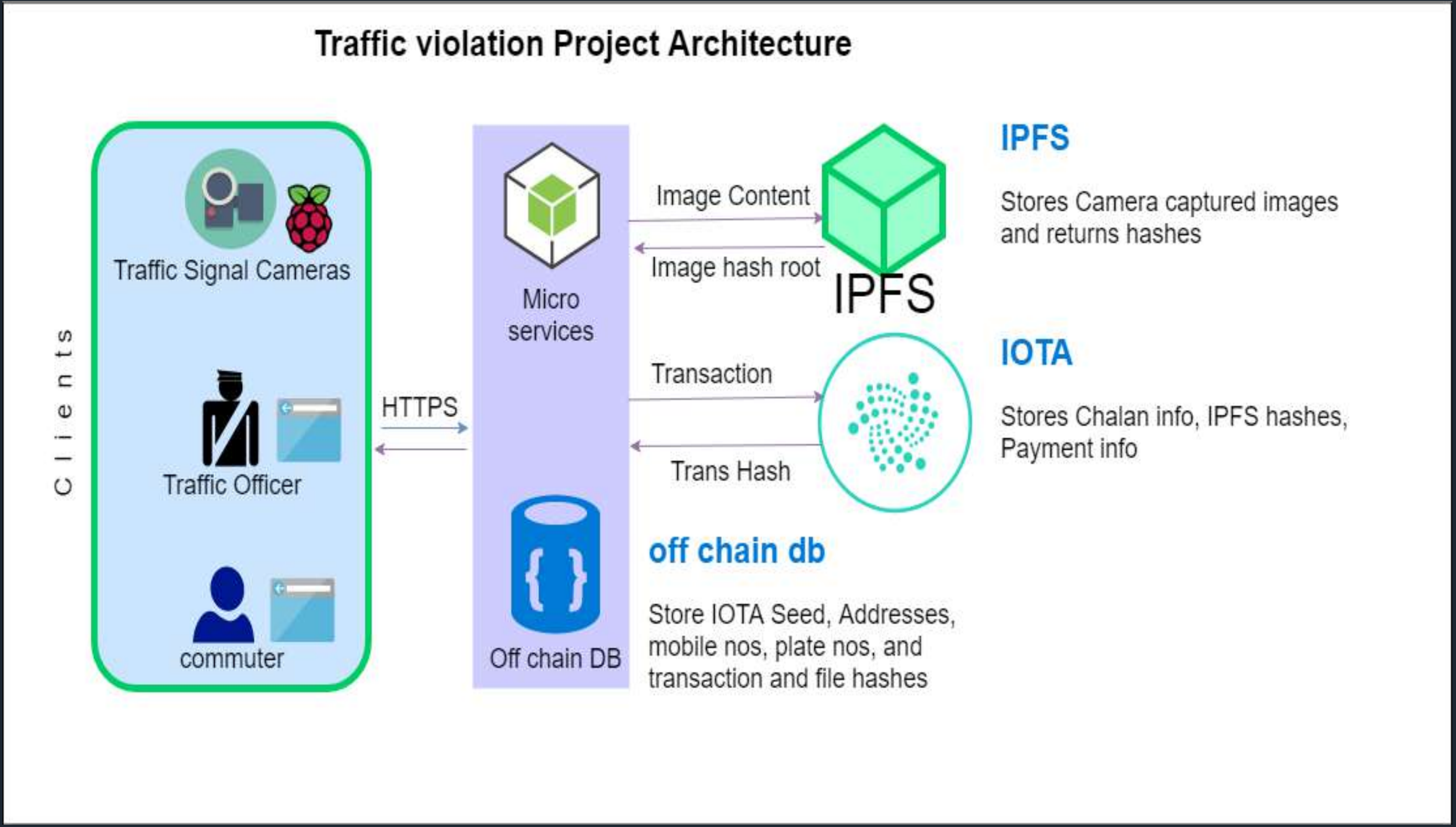


```
const Mam = require('@iota/mam');
const { asciiToTrytes, trytesToAscii } = require('@iota/converter');
var mamState = {}; var seed = '';

const initialize = () => {
    if (seed.length == 0)
        mamState = Mam.init({ provider: 'https://nodes.devnet.thetangle.org:443' });
    else {
        mamState = Mam.init({ provider: 'https://nodes.devnet.thetangle.org:443' }, seed, null);
    }
};
const publish = async (data, isJSON) => {
    try {
        const trytes = asciiToTrytes(isJSON ? JSON.stringify(data) : data);
        const message = Mam.create(mamState, trytes);
        mamState = message.state;
        console.log(message)
        let attachResult = await Mam.attach(message.payload, message.address, 3, 9);
        return attachResult;
    } catch (error) {
        return null;
    }
};
const fetch = async (root, key = "") => {
    try {
        var resp = await Mam.fetch(root,
            (key.length > 0 ? 'restricted' : 'public'), (key.length == 0 ? null : key));
        for (let i = 0; i < resp.messages.length; i++) {
            resp.messages[i] = JSON.parse(trytesToAscii(resp.messages[i]));
        }
        return resp;
    } catch (error) {return null;}
};
```



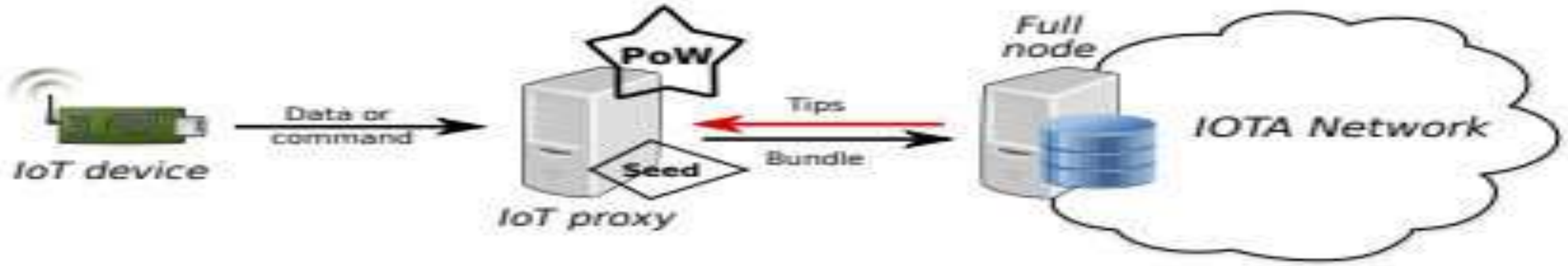
```
const channelize = async () => {
    var msg = { product: "Glass", status: "dispatched from manufacturer" }
    var root = await publish(msg, true);
    msg.status = "Check In at distributor"
    await publish(msg, true);
    msg.status = "dispatched from distributor"
    await publish(msg, true);
    msg.status = "Check In at dealer"
    await publish(msg, true);
    return await root;
}
initialize()
channelize()
fetch('MRXUQGJMKYKZYSKDSIMJCPCXXRPVIVMSIZOXFZTSWOJLHDADNUNJDFVRHDVFGUJQMYTDYZFQVGZOY9KOB')
.then(res => {    console.log(res)})
//https://devnet.thetangle.org/
```

Solution Architecture & Tech Stack

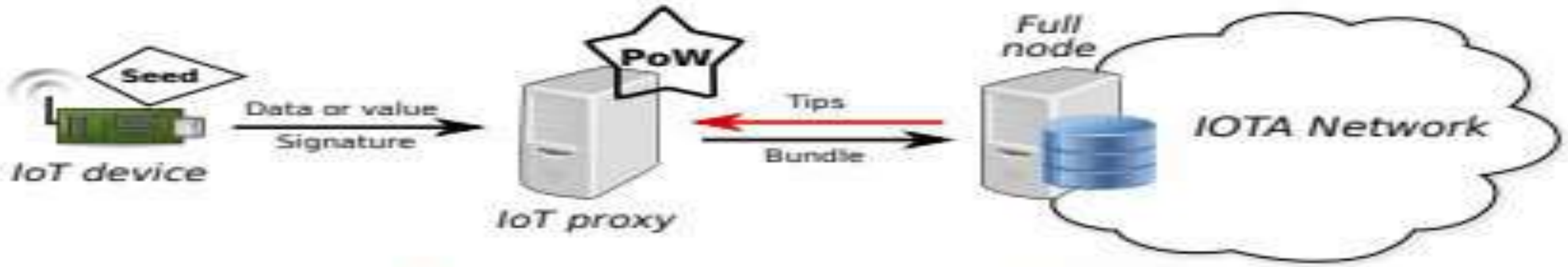

Traffic violation Project Architecture
Clients
Traffic Signal Cameras
Traffic Officer
commuter
HTTPS
Micro services
Off chain DB
Image Content
Image hash root
IPFS
Transaction
Trans Hash
IPFS
Stores Camera captured images and returns hashes
IOTA
Stores Chalan info, IPFS hashes, Payment info
off chain db
Store IOTA Seed, Addresses, mobile nos, plate nos, and transaction and file hashes

Offchain DB	***Mongo DB***
Repository	***GitHub***
Web Design	***angular 7 and angular material and WEB View for android***
Back End	***node.js***
REST services	***express.js***
IOTA	***IOTA.js***
IPFS	***JavaScript***
Agile Project/Task Management	***Trello***
Editor	***Visual studio***

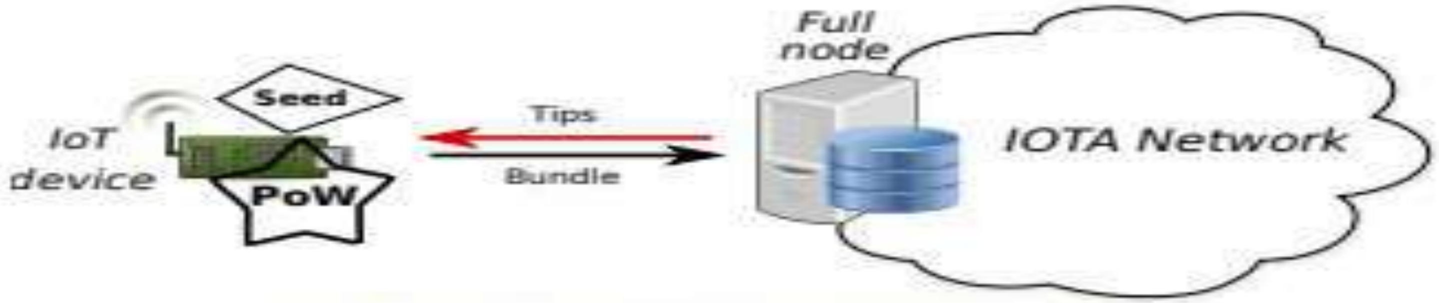
Interaction with Edge Device



(a) Baseline: using IoT proxy for all operations



(b) Using IoT proxy for the Proof-of-Work (PoW)



(c) Direct communication without a proxy

WHO?
which?
when?
WHERE?
which?
WHICH?
WHO?
why?
WHEN?
when?
which?
How?
WHERE?
why?
which?
WHAT?

Meetups Sponsor

nagarro

THINKING BREAKTHROUGHS

Supporters

GOVERNMENT OF TELANGANA

bradsol

SIMPLIFYING BUSINESS

iDEALABS