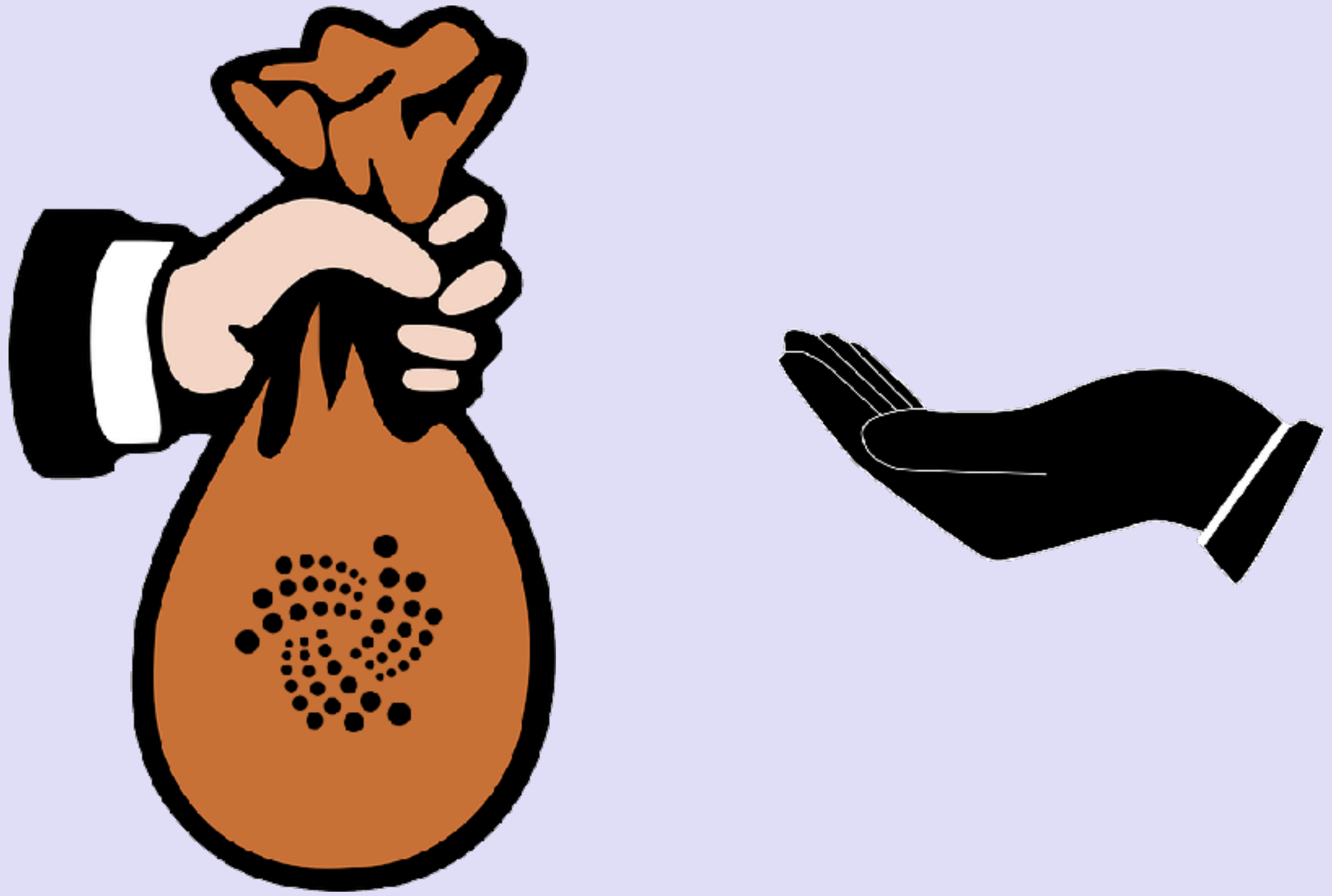
IOTA TUTORIAL 10

Transaction & Bundle



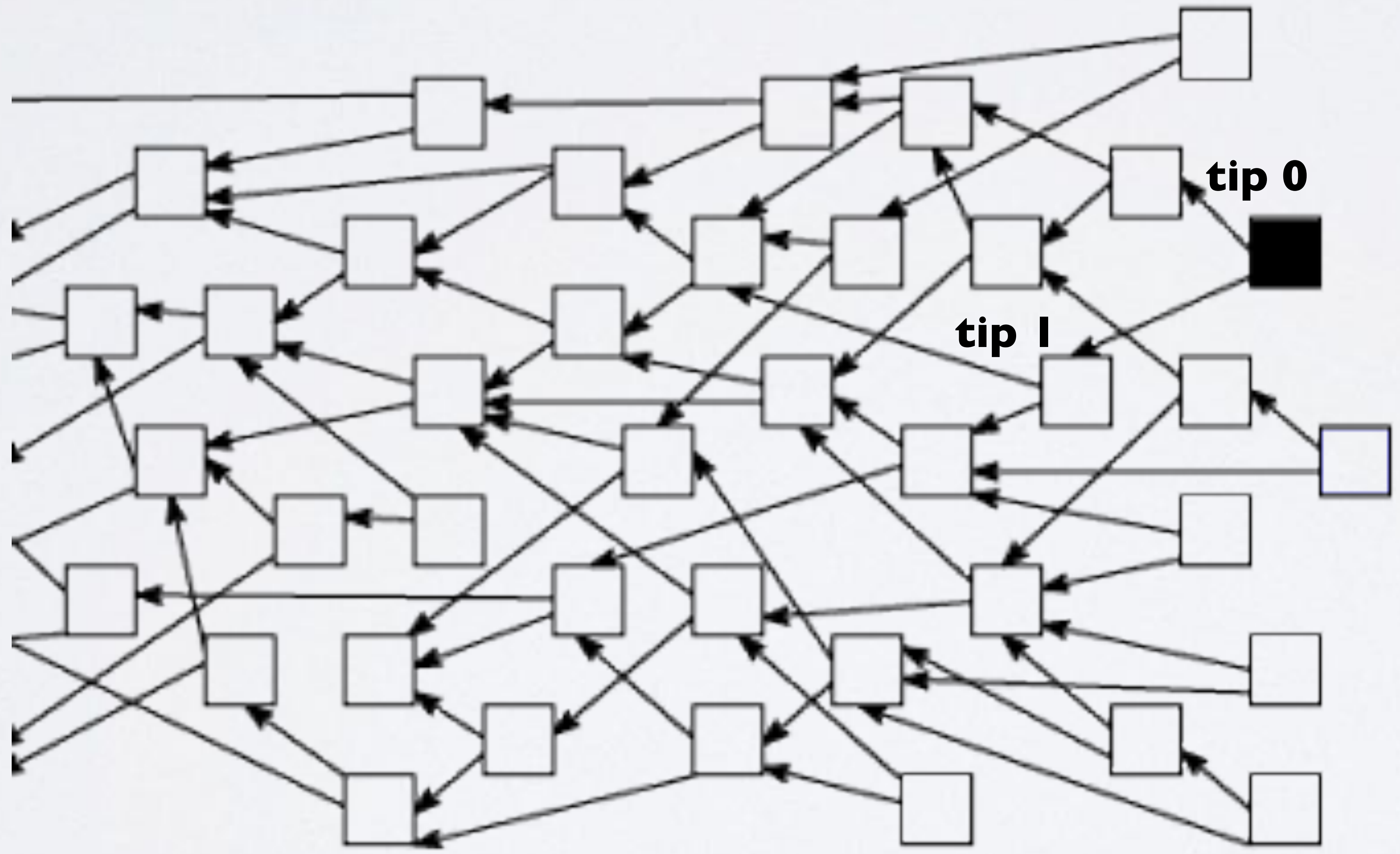
v1.0.0

INTRO

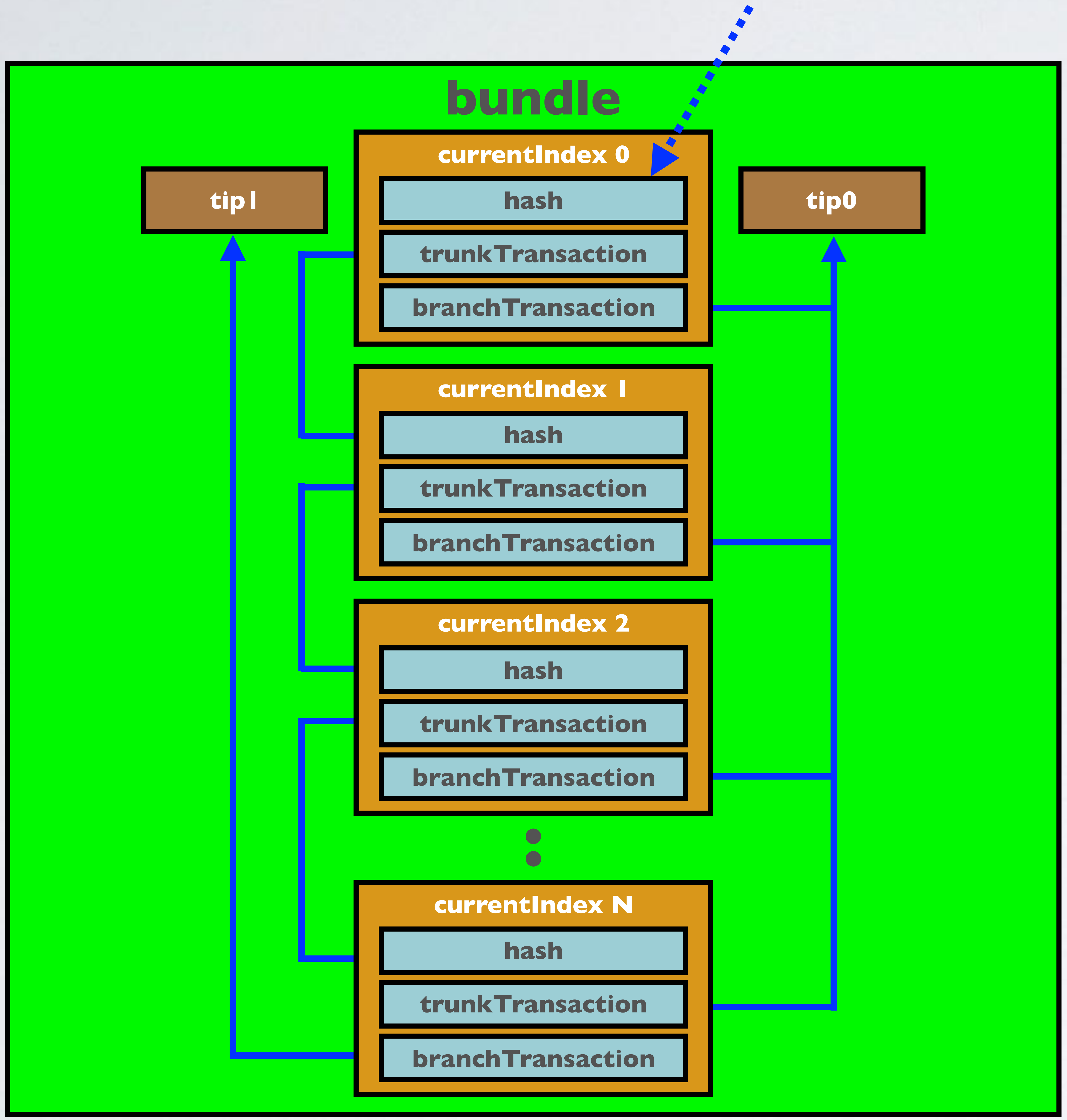
- In this video I will explain how an IOTA transaction looks like and what a transaction bundle is.

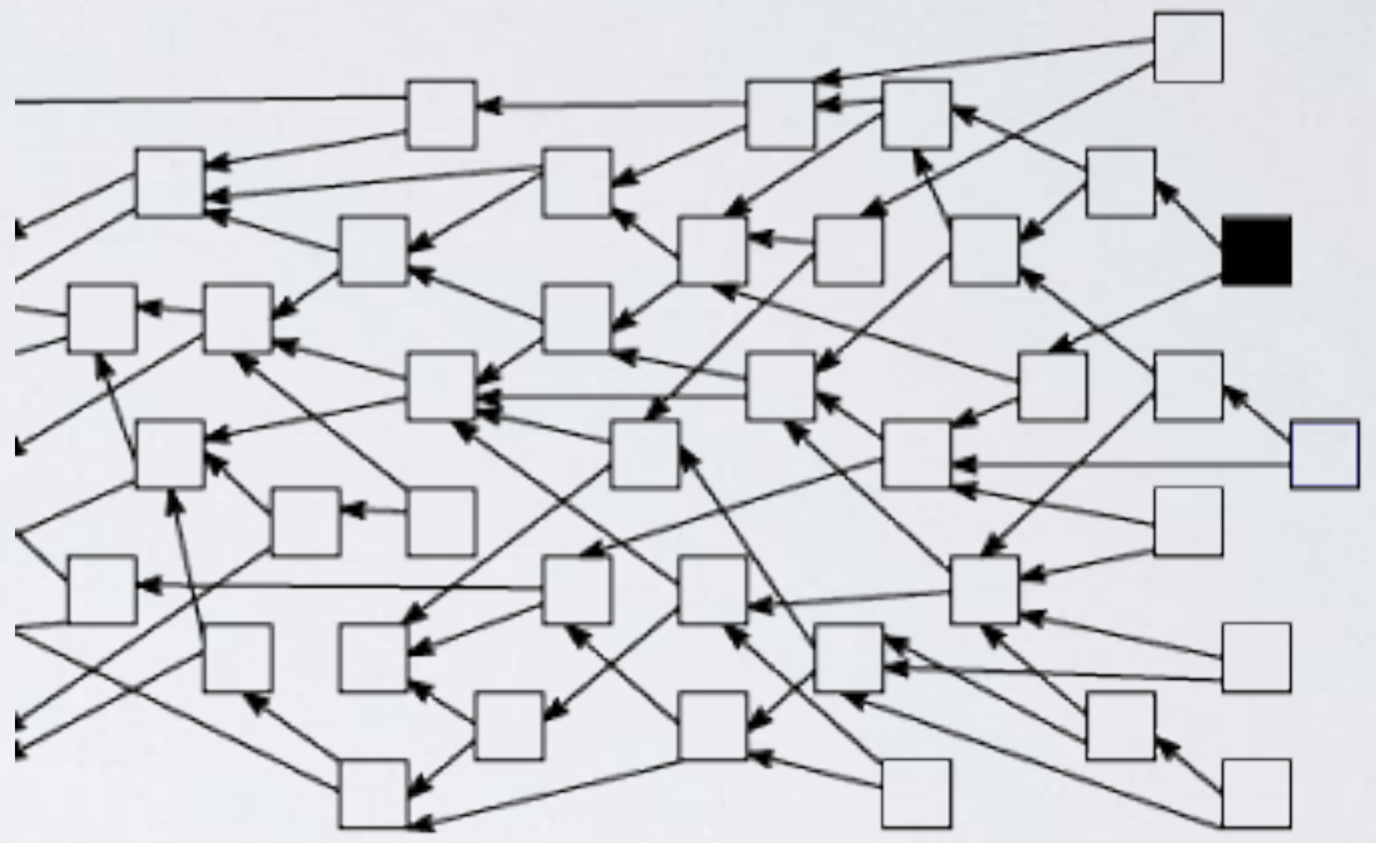
TANGLE

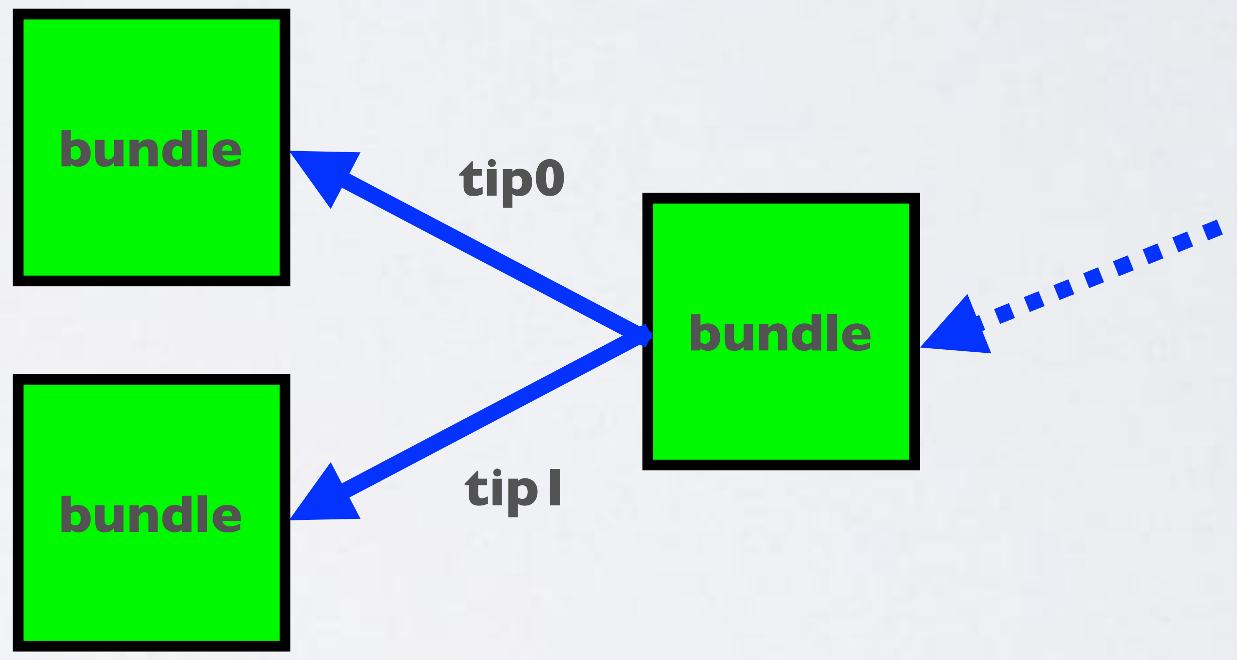
- The squares in the Tangle represents transactions and each NEW transaction should reference transactions which have no other transactions referencing them. These non referenced transactions are called tips (tip 0 and tip 1).
- Each transaction consists of a **bundle** of transactions.



TRANSACTIONS IN BUNDLE

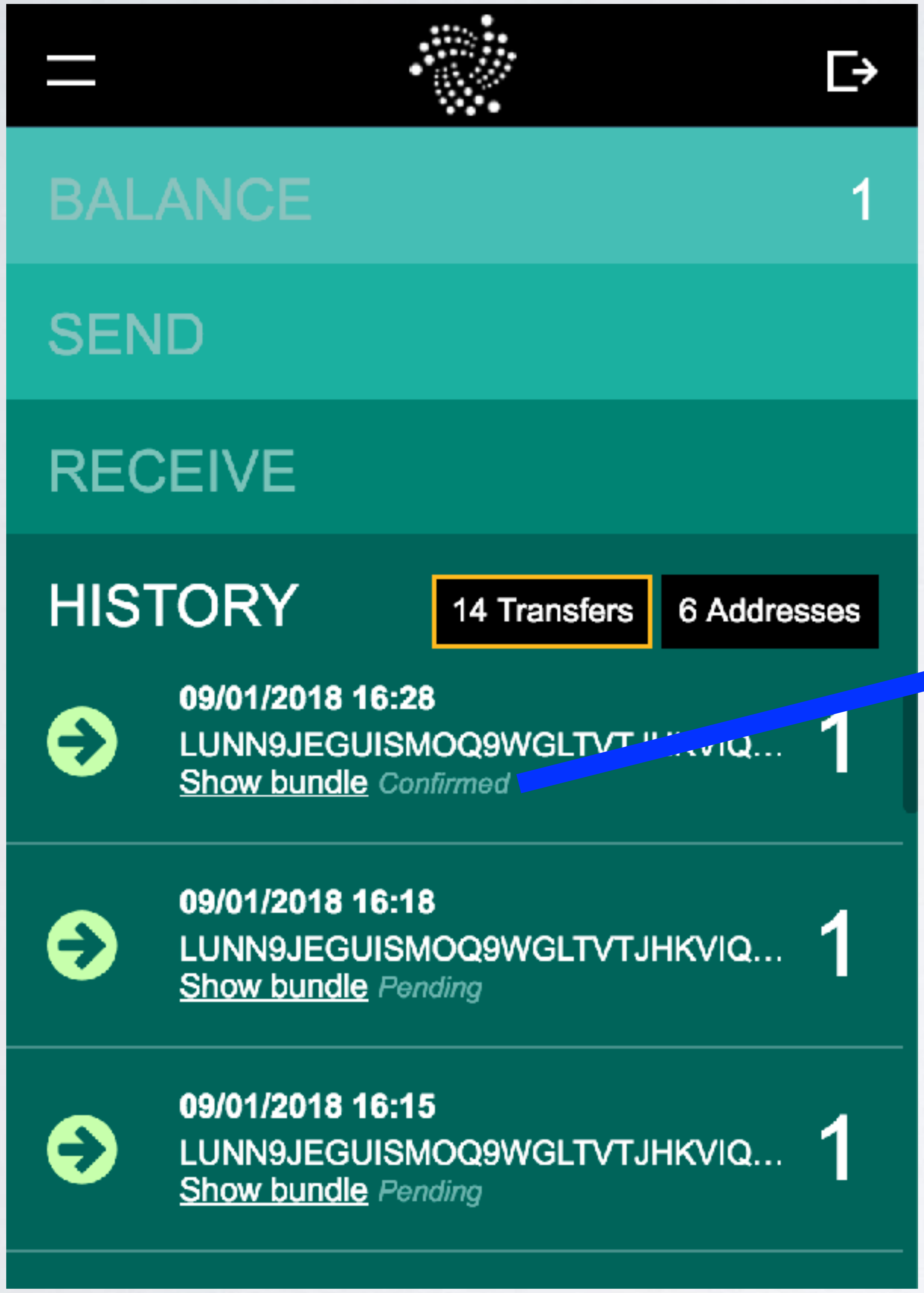






```
currentIndex 0 = tail transaction
currentIndex N = head transaction
```

IOTA LIGHT WALLET

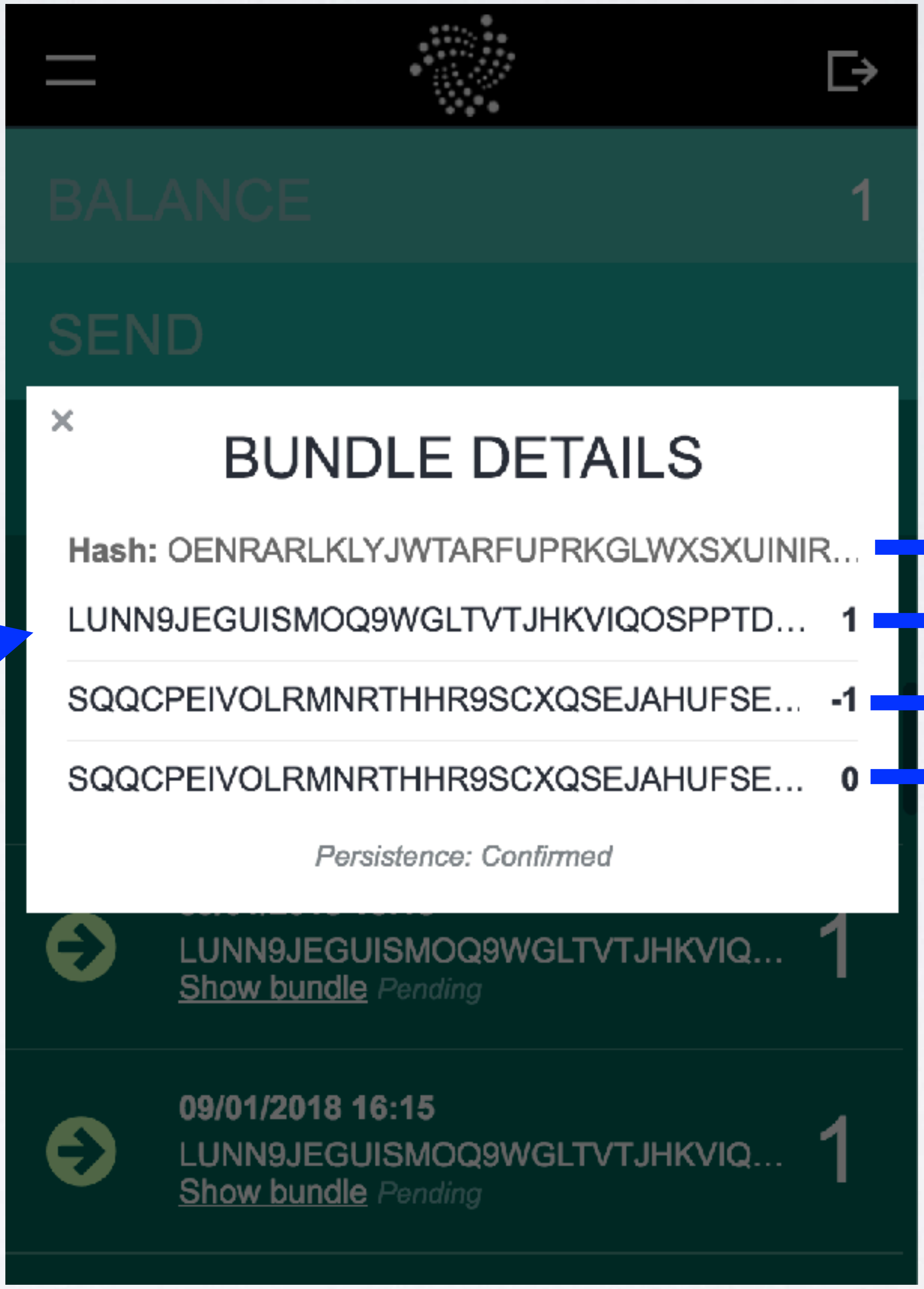

BALANCE 1

SEND

RECEIVE

HISTORY 14 Transfers 6 Addresses

09/01/2018 16:28	LUNN9JEGUISMOQ9WGLTVTJHKVIQ... Show bundle Confirmed	1
09/01/2018 16:18	LUNN9JEGUISMOQ9WGLTVTJHKVIQ... Show bundle Pending	1
09/01/2018 16:15	LUNN9JEGUISMOQ9WGLTVTJHKVIQ... Show bundle Pending	1


BUNDLE DETAILS

Hash: OENRARLKLYJWTARFUPRKGLWXSXUINIR... — TX0 hash

LUNN9JEGUISMOQ9WGLTVTJHKVIQOSPPTD...	1	TX0 address and value
SQQCPEIVOLRMNRTHHR9SCXQSEJAHUFSE...	-1	TX1 address and value
SQQCPEIVOLRMNRTHHR9SCXQSEJAHUFSE...	0	TX2 address and value

Persistence: Confirmed

TRANSACTIONS IN BUNDLE

- An example of an IOTA transaction where 3 IOTA's are transferred from Alice's address HRKD…XKHX to Bob's address JHYL…HTUZ
  https://www.mobilefish.com/download/iota/transactions_in_bundle_example1.txt

- This transaction consists of a bundle of transactions: currentIndex 0 refers to transaction 0, currentIndex 1 refers to transaction 1, etc..

- All these transactions have the same bundle hash:
  UMGX…LQVB

- All transactions in the same bundle should be treated as an atomic unit.
  It means that either all transactions of a bundle are confirmed, or none of them are confirmed.

TRANSACTIONS IN BUNDLE

- Every transaction in the bundle requires it own PoW, see the different nonces in the transactions.
- Full nodes stores every transaction that they and their neighbours are aware of, which means all transactions in the bundle.
- Light nodes don't store anything, they are stateless. Light nodes use API calls to the full node they are connected to, to get the information such as addresses and balances.
- After a snapshot all meta data such as tags and messages are deleted.
  Only addresses with positive balances are restored by the full nodes.

TRANSACTIONS IN BUNDLE

- There can be 3 different types of transactions inside a bundle:
  - **Output transactions**
    Transactions where IOTA's are send to one or multiple addresses.
    The IOTA light wallet can only send to one address. These transactions are easily recognised because the transaction value is always greater than 0 and the address does not belong the sender.

TRANSACTIONS IN BUNDLE

- **Input transactions**
  There are two types of input transactions:

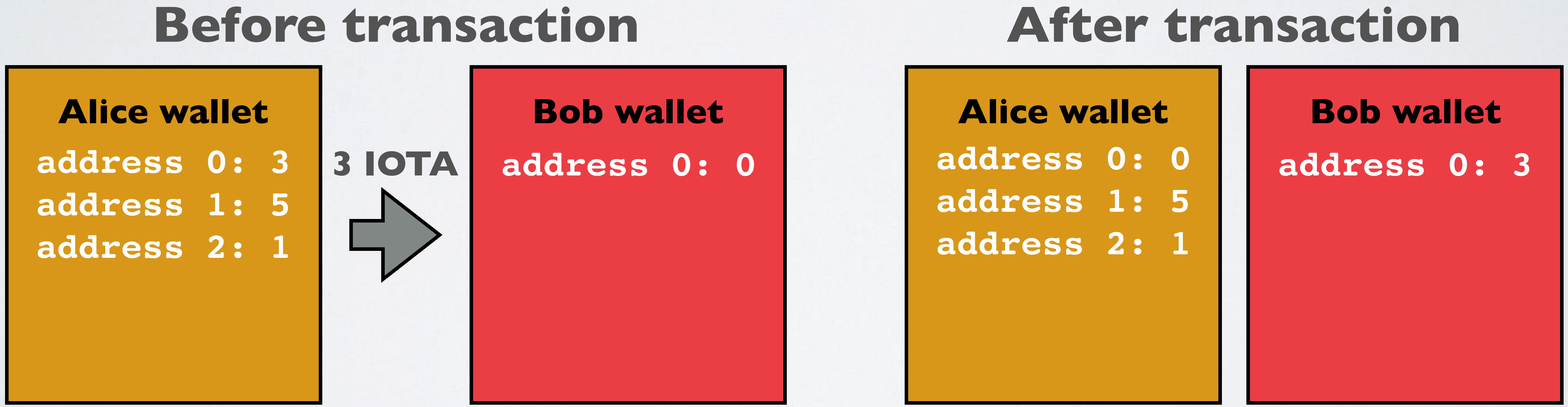
  Input transactions where the value is negative.
  These are the transactions where the complete balance from that address is spent.

  Input transactions where the value is greater than 0.
  These are the transactions where unspent/not used IOTA's are send to a new change address in the senders wallet.

- **Meta transactions**
  Zero value transactions are meta transactions. The signatureMessageFragment of these transactions could either hold a signature or a message fragment.

TRANSACTION EXAMPLE 1

- Alice wants to send 3 IOTAs to Bob. Alice's wallet starts with address 0 and adds the balances of the consecutively addresses until 3 IOTAs are reached or exceeded. Any extra amount over the payment amount will be sent to a new address called the change address, which means you will not have to worry about address reuse in typical cases.

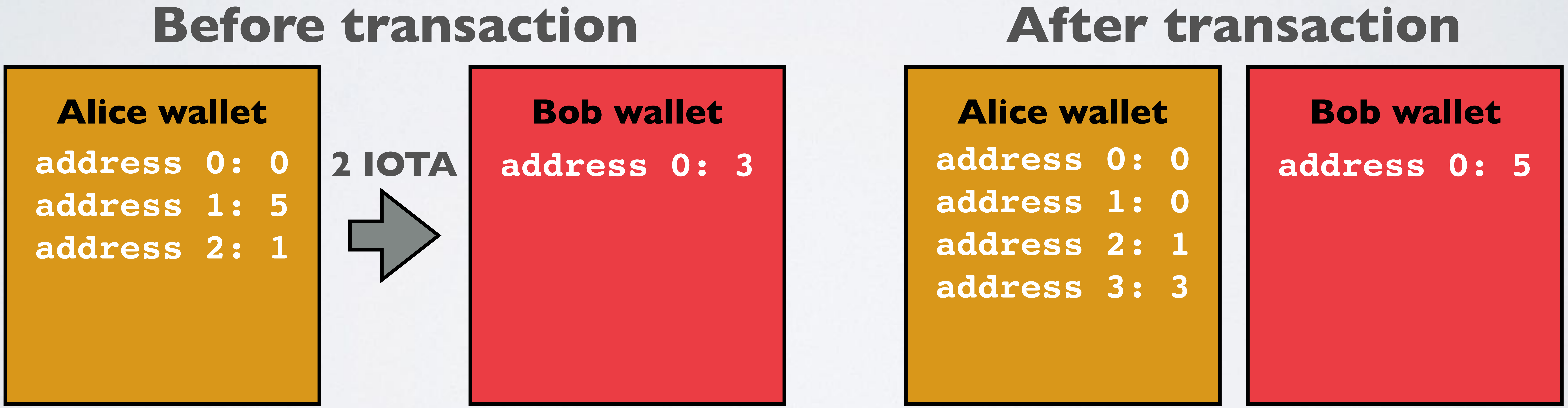


TRANSACTION EXAMPLE 1

- See example 1 (using iota.lib.js v0.4.6): https://www.mobilefish.com/download/iota/transactions_in_bundle_example1.txt

TRANSACTION EXAMPLE 2

- Alice wants to send 2 IOTA's to Bob. Alice's wallet starts with address 0 and adds the balances of the consecutively addresses until 2 IOTA's are reached or exceeded. Any extra amount over the payment amount will be sent to a new address called the change address, which means you will not have to worry about address reuse in typical cases.

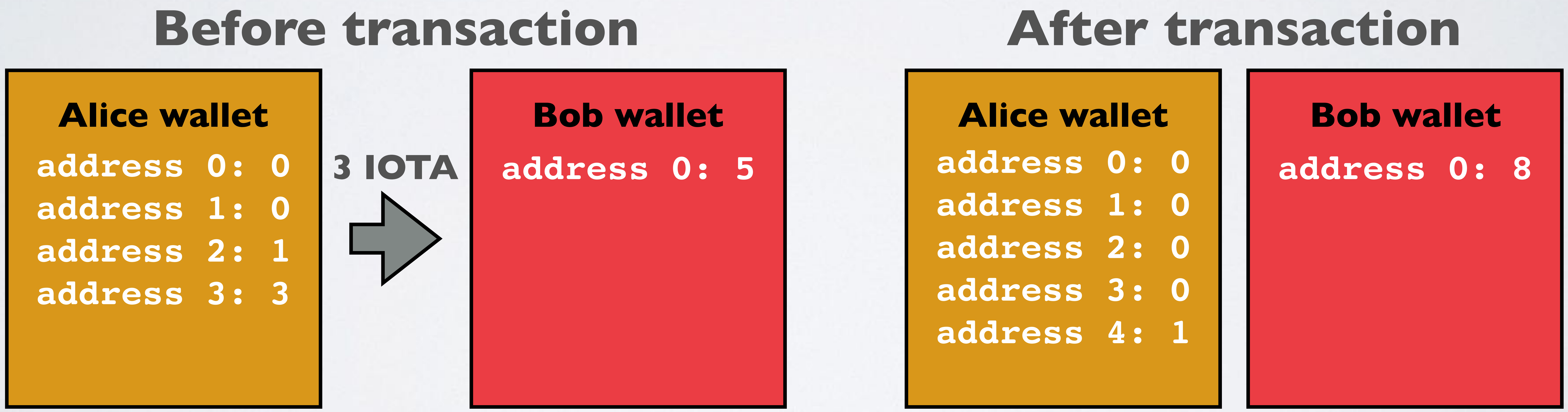


TRANSACTION EXAMPLE 2

- See example 2 (using iota.lib.js v0.4.6): https://www.mobilefish.com/download/iota/transactions_in_bundle_example2.txt

TRANSACTION EXAMPLE 3

- Alice wants to send 3 IOTA's to Bob. Alice's wallet starts with address 0 and adds the balances of the consecutively addresses until 3 IOTA's are reached or exceeded. Any extra amount over the payment amount will be sent to a new address called the change address, which means you will not have to worry about address reuse in typical cases.



Before transaction

Alice wallet

```
address 0: 0
address 1: 0
address 2: 1
address 3: 3
```

3 IOTA

Bob wallet

```
address 0: 5
```

After transaction

Alice wallet

```
address 0: 0
address 1: 0
address 2: 0
address 3: 0
address 4: 1
```

Bob wallet

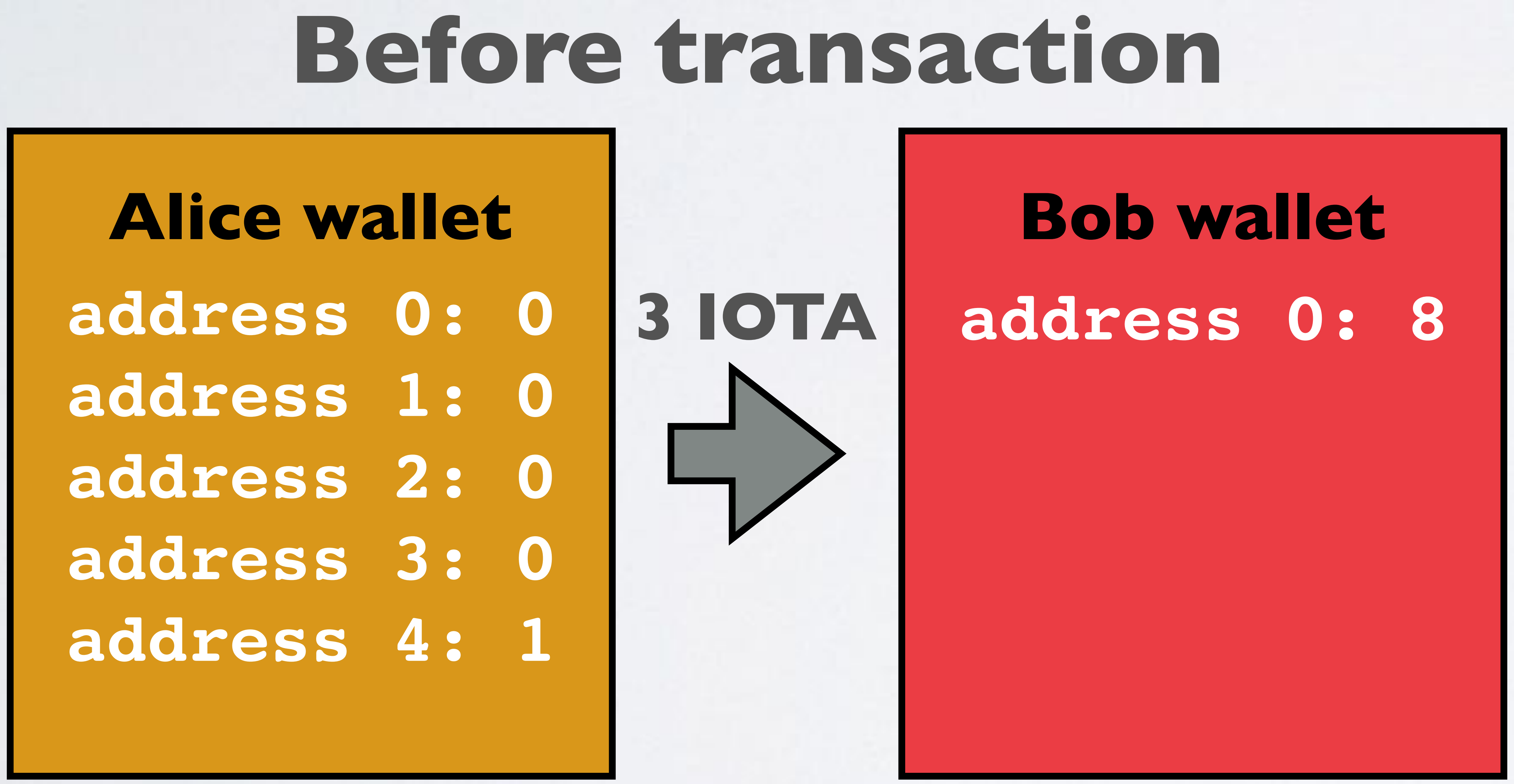
```
address 0: 8
```

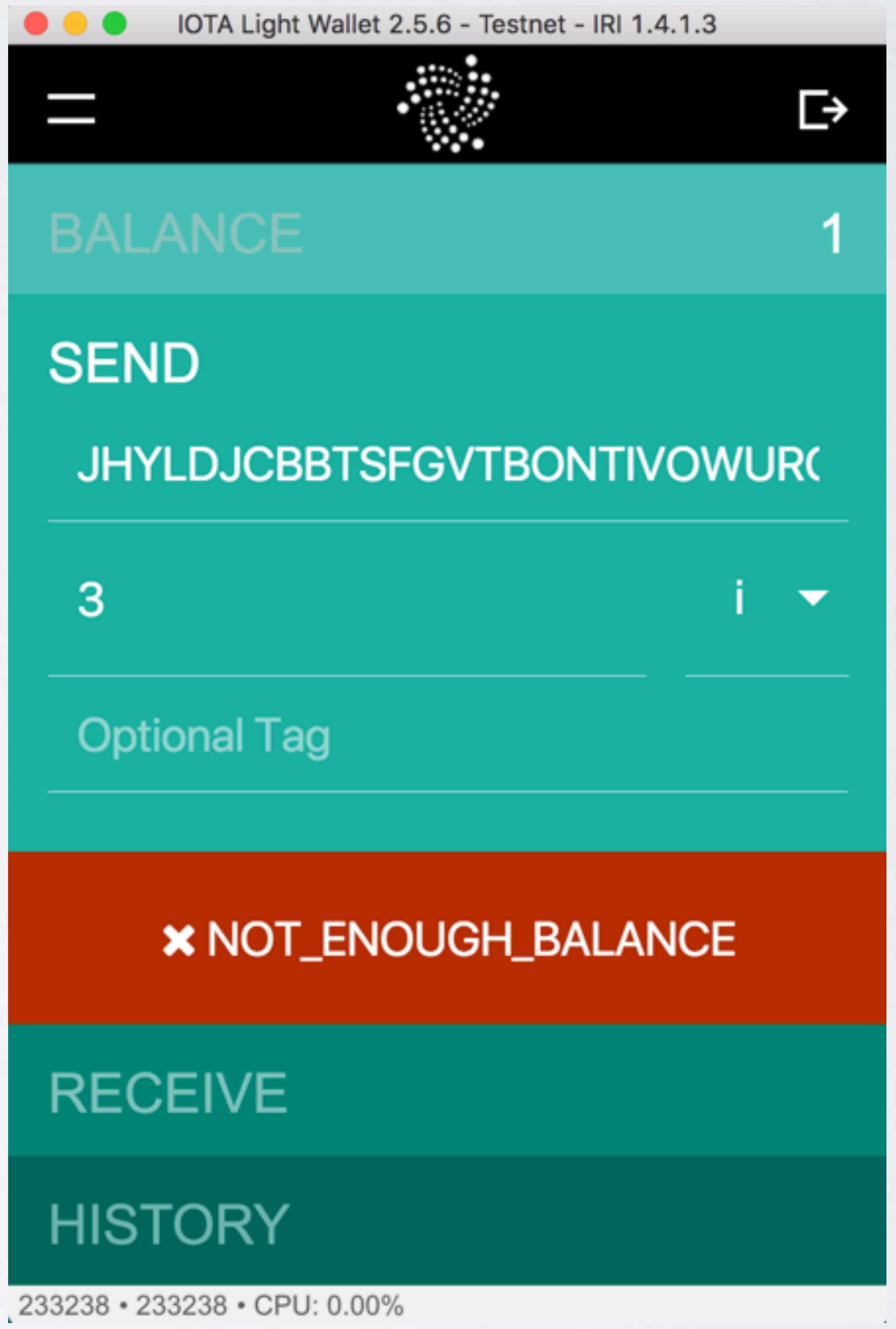
TRANSACTION EXAMPLE 3

- See example 3 (using iota.lib.js v0.4.6): https://www.mobilefish.com/download/iota/transactions_in_bundle_example3.txt

TRANSACTION EXAMPLE 4

- Alice wants to send 3 IOTA's to Bob. Alice's wallet shows a warning message because her total balance is insufficient.




IOTA Light Wallet 2.5.6 - Testnet - IRI 1.4.1.3

BALANCE 1

SEND

JHYLDJCBBTSFGVTBONTIVOWUR(

3 i

Optional Tag

✖ NOT_ENOUGH_BALANCE

RECEIVE

HISTORY

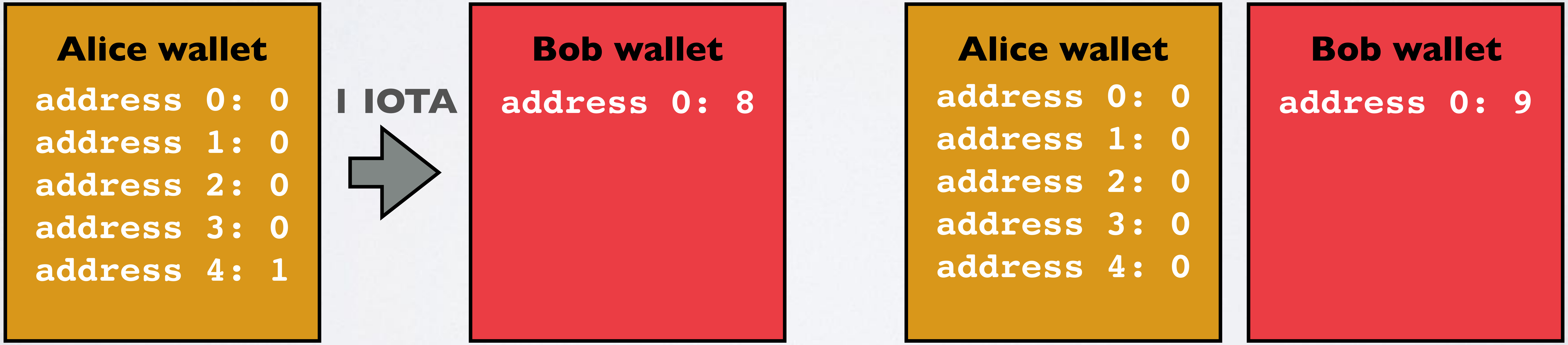
233238 • 233238 • CPU: 0.00%

TRANSACTION EXAMPLE 5

- Alice wants to send 1 IOTA to Bob. Alice's wallet starts with address 0 and adds the balances of the consecutively addresses until 1 IOTA is reached or exceeded. Any extra amount over the payment amount will be sent to a new address called the change address, which means you will not have to worry about address reuse in typical cases.

Before transaction

After transaction



TRANSACTION EXAMPLE 5

- See example 5 (using iota.lib.js v0.4.6): https://www.mobilefish.com/download/iota/transactions_in_bundle_example5.txt

TRANSACTION OBJECT

- An example of an IOTA transaction where 3 IOTA's are transferred from Alice's address HRKD…XKHX to Bob's address JHYL…HTUZ
  https://www.mobilefish.com/download/iota/transactions_in_bundle_example1.txt

- In this example you see an array of 3 transaction objects all having the same bundle hash. A transaction object is formatted using JSON (JavaScript Object Notation).

- In the following slides each transaction object key will be explained.

TRANSACTION OBJECT (IOTA.LIB.JS 0.4.6)

Key Name	Type	Tryte size	Description
hash	String	81	**Uniquely identify the transaction on the Tangle.** This value is generated by taking a hash of the raw transaction trits.
signatureMessage Fragment	String	2187	**Holds either a signature or a message, which may be fragmented over multiple transactions in a bundle.** If value < 0, this value contains a fragment of the signature authorising the spending of the IOTAs. If value > 0, this value is an (optional) string message attached to the transaction. If value = 0, this value could be either a signature or message fragment, depending on the previous transaction. In case of a spent input, the signature is stored. In case of a message value, no signature is required. If there is no message, the field contains all 9's.

TRANSACTION OBJECT (IOTA.LIB.JS 0.4.6)

Key Name	Type	Tryte size	Description
address	String	81	**The address associated with this transaction.** If value field >0, the field contains the recipient address. If value field <=0, the field contains the sender's address (= withdrawal address)
value	Int	-	**The number of IOTA's being transferred in this transaction:** If this value is negative, then the address is spending IOTA's. If it is positive, then the address is receiving IOTA's. If it is zero, then this transaction is being used to carry metadata (such as a signature fragment or a message fragment) instead of transferring IOTA's.
obsoleteTag	String	27	**User-defined tag (will be removed soon)**
timestamp	Int	-	**Unix timestamp when the transaction was issued.** Devices can specify any timestamp when creating transactions, so this value is not safe to use for security measures such as resolving double-spends. Timestamps in IOTA are not enforced.

TRANSACTION OBJECT (IOTA.LIB.JS 0.4.6)

Key Name	Type	Tryte size	Description
currentIndex	Int	-	**The transaction's position in the bundle.** If currentIndex == 0, the tx is called the "tail transaction". If currentIndex == lastIndex, the tx is called the "head transaction".
lastIndex	Int	-	**The last transaction position in the bundle.** The total number of transactions in the bundle: (lastIndex + 1) This value is attached to every transaction to make it easier to traverse and verify bundles.
bundle	String	81	**The bundle hash identifies which transactions belongs to the same bundle.** This value is generated by taking a hash of the metadata from all transactions in the bundle.

TRANSACTION OBJECT (IOTA.LIB.JS 0.4.6)

Key Name	Type	Tryte size	Description
trunkTransaction	String	81	**Tip 0 hash or next transaction hash.** If this transaction is not the head transaction, the trunkTransaction will hold the hash of the next transaction of the bundle (currentIndex + 1). If this transaction is the head transaction, the trunkTransaction will hold the tip 0 hash.
branchTransaction	String	81	**Tip 0 hash or tip 1 hash.** If this transaction is not the head transaction, the branchTransaction will hold tip 0 hash. If this transaction is the tail transaction, the branchTransaction will hold tip 1 hash.
tag	String	27	**User-defined tag to easily find a transaction.** A tag is used to classify a transaction and helps you to find your transactions using a Tangle explorer. Many transactions have empty tags. If there is no tag, the field contains all 9's.

TRANSACTION OBJECT (IOTA.LIB.JS 0.4.6)

Key Name	Type	Tryte size	Description
attachment Timestamp	Int	-	**Timestamp after PoW** The time stamp for when PoW is completed.
attachment Timestamp LowerBound	Int	-	**Attachment Time (lower bound)** Is a slot for future use.
attachment Timestamp UpperBound	Int	-	**Attachment Time (upper bound)** Is a slot for future use.
nonce	String	27	**The Proof of Work solution.** The nonce is required for the transaction to be accepted by the network. It is generated by doing Proof of Work (either in IRI via the attachToTangle API call, or with one of the libraries such as CCURL).
persistence	Bool	-	**Indicates if the transaction is pending or confirmed.** If true, transaction is confirmed. If false, transaction is pending.

TRANSACTION OBJECT (IOTA.LIB.JS 0.4.6)

- The keys used in the transaction object are not finalised and may change in the future.

```
{
    "hash": "ZXKI...9999",
    "signatureMessageFragment": "9999...9999",
    "address": "JHYL...HTUZ",
    "value": 3,
    "obsoleteTag": "OA99...9999",
    "timestamp": 1515494426,
    "currentIndex": 0,
    "lastIndex": 2,
    "bundle": "UMGX...OLQVB",
    "trunkTransaction": "YDDQ...9999",
    "branchTransaction": "DOX...X999",
    "tag": "999999999999999999999999999",
    "attachmentTimestamp": 1515496588388,
    "attachmentTimestampLowerBound": 0,
    "attachmentTimestampUpperBound": 3812798742493,
    "nonce": "EA999RL99999999999999999999",
    "persistence": true
}
```

SIGNATURE STORAGE AT DIFFERENT SECURITY LEVELS

- In IOTA tutorial 9 I have explained what security levels are.
- There are 3 security levels: 1, 2 and 3. By increasing the security level you increase the key size, meaning you increase the private key size.
- When a value transaction is created the complete balance from one or more addresses are spent (value < 0). These transaction objects signatureMessageFragment fields contains the signature. Depending on the key size, this signature is fragmented and stored in 1, 2 or 3 transactions.
- This means:
  If you use security level 1, the signature is stored in 1 transaction.
  If you use security level 2, the signature is split up across 2 transactions.
  If you use security level 3 the signature is split up across 3 transactions.

SIGNATURE STORAGE EXAMPLES

- Alice has 3 different wallets.
  Wallet 1, using security level 1 containing only 1 IOTA at address 0.
  Wallet 2, using security level 2 containing only 1 IOTA at address 0.
  Wallet 3, using security level 3 containing only 1 IOTA at address 0.

- Alice makes a tx from wallet 1, 1 IOTA is send to address X (using iota.lib.js v0.4.6):
  https://www.mobilefish.com/download/iota/transactions_in_bundle_security_level1.txt

- Alice makes a tx from wallet 2, 1 IOTA is send to address X (using iota.lib.js v0.4.6):
  https://www.mobilefish.com/download/iota/transactions_in_bundle_security_level2.txt

- Alice makes a tx from wallet 3, 1 IOTA is send to address X (using iota.lib.js v0.4.6):
  https://www.mobilefish.com/download/iota/transactions_in_bundle_security_level3.txt

SIGNATURE STORAGE AT DIFFERENT SECURITY LEVELS

- Wallet 1, using security level 1, the signature is stored in 1 transaction.
  Wallet 2, using security level 2, the signature is fragmented and stored in 2 transactions.
  Wallet 3, using security level 3, the signature is fragmented and stored in 3 transactions.

- By increasing the security level you increase the signature size and thus the number of transactions needed to store the signature.

- IOTA signatures are larger than Bitcoin signatures due to IOTA's use of Winternitz one-time signatures to gain quantum resistance.

TRANSACTION SIZE

- Each single transaction inside a bundle consists of 2673 trytes and much of it is taken by the signatureMessageFragment which has a size of 2187 trytes (approx 82%).

- Algorithm to convert bits to trits:
  $\text{trits} = \text{bits} \times \log_e(2) / \log_e(3) = \text{bits} \times \ln(2) / \ln(3)$

- Algorithm to convert trits to bits:
  $\text{bits} = \text{trits} \times \log_e(3) / \log_e(2) = \text{trits} \times \ln(3) / \ln(2)$

- Convert 2673 trytes to bytes:
  $\text{bytes} = (\text{trytes} \times 3 \times \ln(3) / \ln(2)) / 8 = (2673 \times 3 \times \ln(3) / \ln(2)) / 8 = \sim 1589 =$ 1.55 kBytes

TRANSACTION SIZE

- **A single transaction inside a bundle requires 2673 trytes or ~1.55 kBytes.**

HOW MANY TRANSACTIONS IN A BUNDLE

- A bundle can have one transaction, for example when you attach an address to the Tangle.
- A bundle can have an X number of transactions.
- An example:
  - Alice has a wallet (using security level 2) with address 0 to address 99 with each address having one IOTA. When Alice transfers her complete wallet balance to Bob, she creates a transaction bundle containing 201 transactions: 1 transaction to Bob, 100 transactions withdrawing 1 IOTA from each address and 100 meta transactions to store the second signature fragment.

HOW MANY TRANSACTIONS IN A BUNDLE

- Each transaction inside the bundle requires a PoW.
  Let assume a PoW takes 20 seconds per transaction.
  Please note: **This 20 sec is arbitrary chosen!**
  In this example the total time to create the bundle, takes:
  201 transactions x 20 sec / tx = 67 minutes.

- You can easily create a transaction bundle containing lots of transactions.

- However watch out for the Proof of Work.