IOTA TUTORIAL 27

Why normalizedBundleHash?
Why not reuse an address for outgoing txs?

INTRO

- In IOTA tutorial 6 I have explained why you should not reuse an address for outgoing transactions by using the Lamport One Time Signature scheme.
  That was a simplified explanation but not an accurate one.
  This tutorial will provide you the correct answer.

- In IOTA tutorial 16 I have never explained why the bundleHash is normalized.
  In this tutorial I will explain why this it is needed.

PREREQUISITES

- I assume that you have watched:
  - IOTA tutorial 8: Cryptographic sponge construction
  - IOTA tutorial 9.1: Key, Digests & Address
  - IOTA tutorial 10: Transaction and bundle
  - IOTA tutorial 15: BundleHash
  - IOTA tutorial 16: normalizedBundleHash
  - IOTA tutorial 17: Create and validate a signature

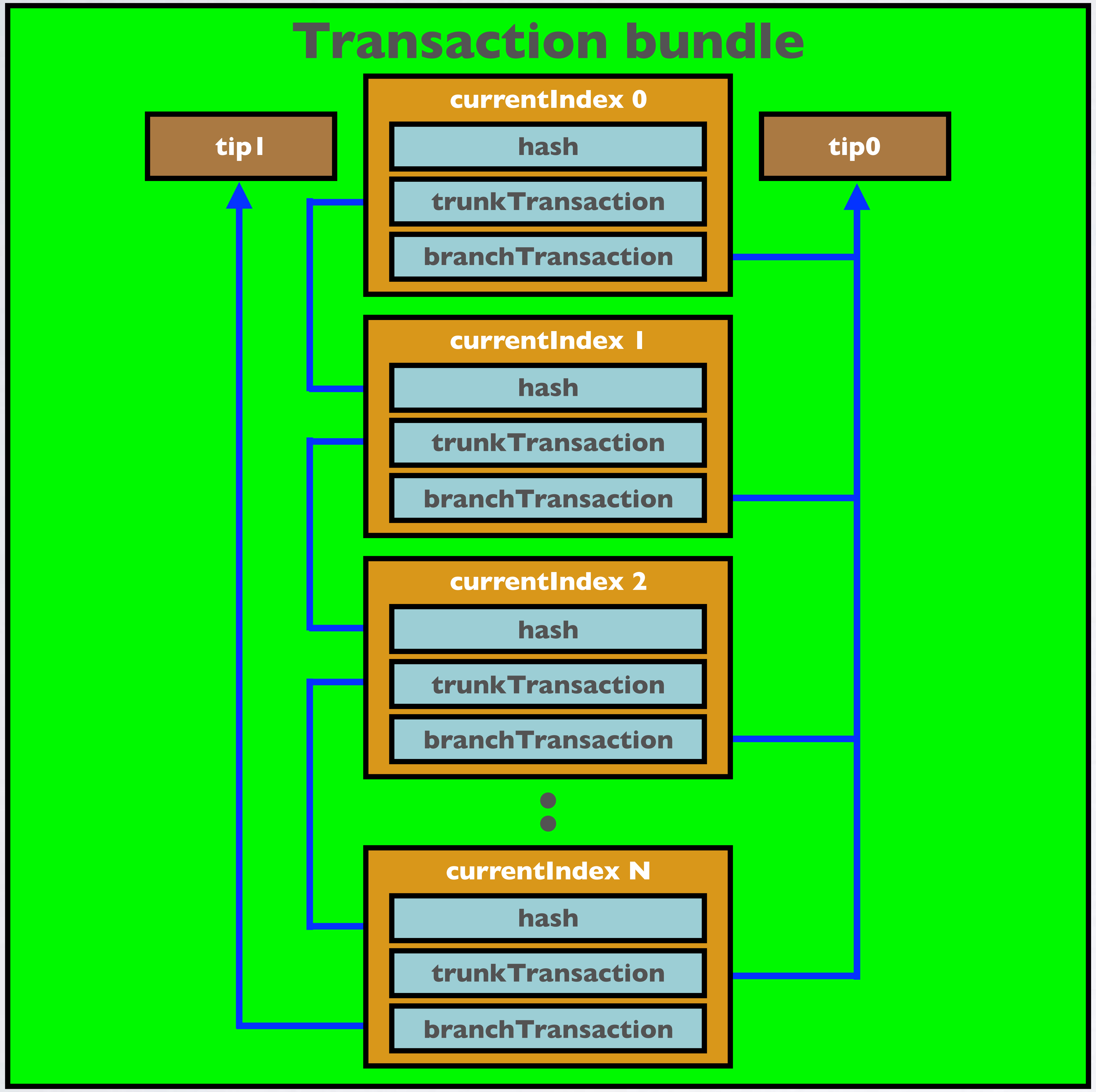
PREREQUISITES

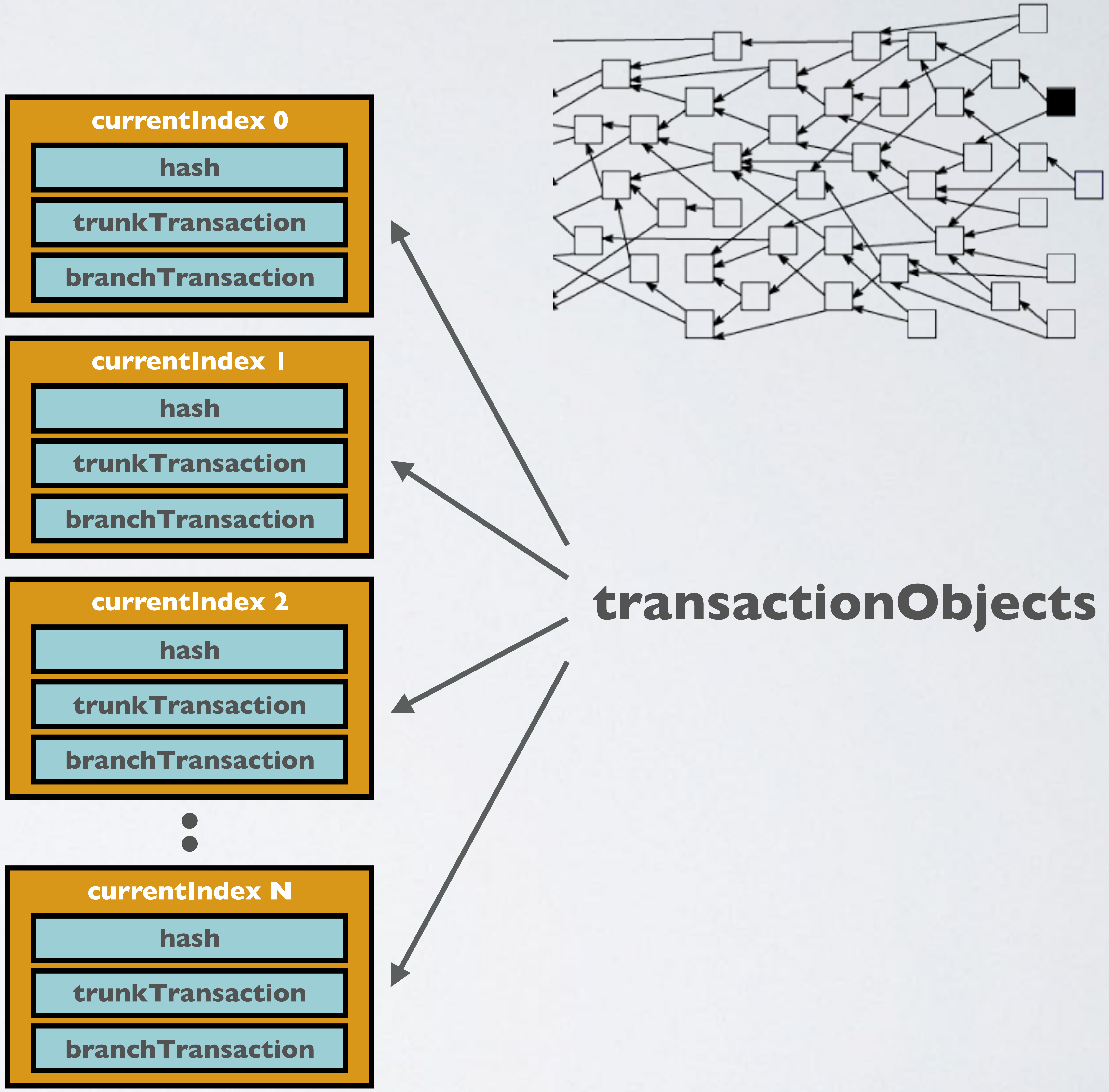
- If you have not watched these videos you probably will not understand this tutorial. I highly recommended that you first watch these tutorials.

QUICK REFRESHER

- To be on the same page, in the following slides I will give you a quick refresher:
  - What is a transaction bundle and transaction objects.
  - What is a bundleHash and how it is created.
  - What is a normalizedBundleHash and how it is created.
  - How to calculate the number of hashes.
  - How to create and validate a signatureFragment.
  - How is an address calculated.

TRANSACTIONBUNDLE





TRANSACTIONOBJECT EXAMPLE

- This is what a single transactionObject looks like in a transaction bundle:

```
{
    "hash": "YDDQ...A9999",
    "signatureMessageFragment": "JHAK...MVGY",
    "address": "HRKD...XKHX",
    "value": -3,
    "obsoleteTag": "999999999999999999999999999",
    "timestamp": 1515494426,
    "currentIndex": 1,
    "lastIndex": 2,
    "bundle": "RTGX...LQCY",
    "trunkTransaction": "WVCLP...99999",
    "branchTransaction": "DOXV...X999",
    "tag": "999999999999999999999999999",
    "attachmentTimestamp": 1515496571334,
    "attachmentTimestampLowerBound": 0,
    "attachmentTimestampUpperBound": 3812798742493,
    "nonce": "AZ999IOB99999999999999999",
    "persistence": true
},
```

How is this bundleHash created?

BUNDLEHASH

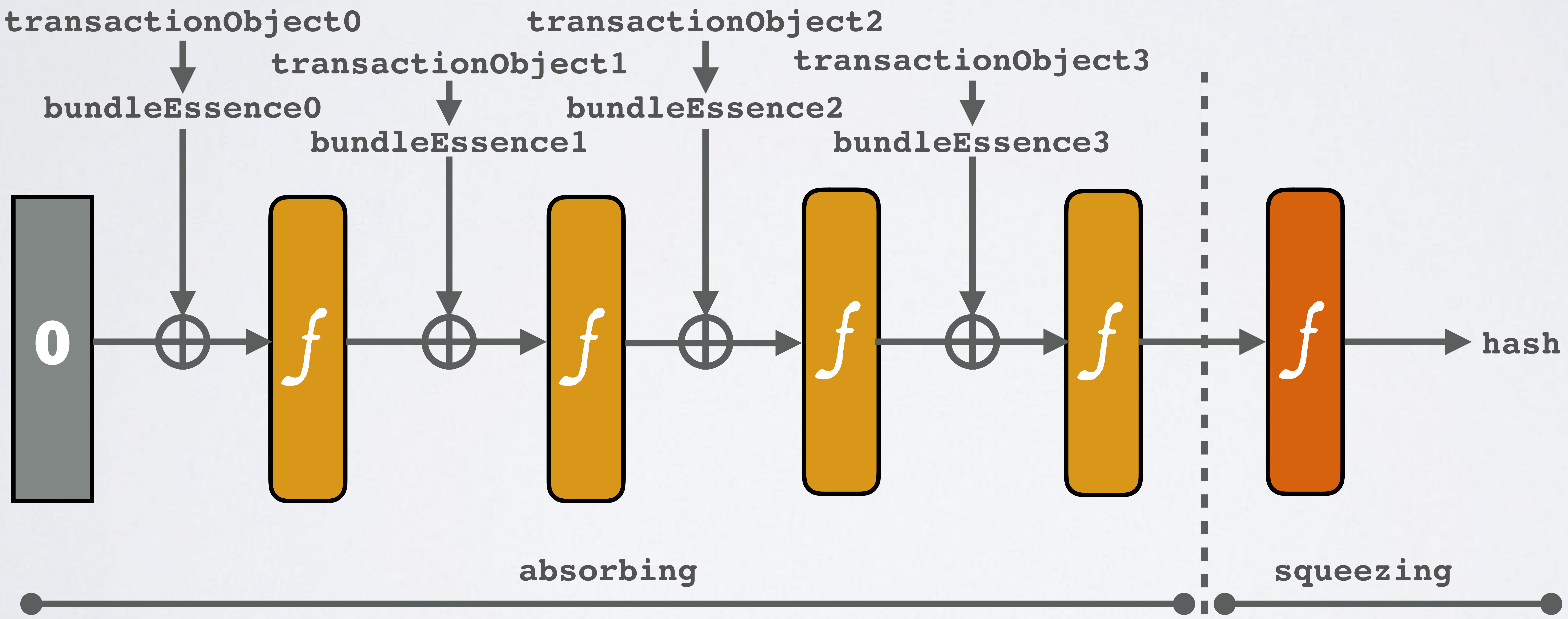
- The bundle transactionObject addresses, values, obsoleteTags, timestamps, currentIndexes and lastIndexes are used to calculate the bundleEssences:

```
bundleEssence =
convertToTrits(address) +
convertToTrytes(valueTrits) +
obsoleteTag +
convertToTrytes(timestampTrits) +
convertToTrytes(currentIndexTrits) +
convertToTrytes(lastIndexTrits))
```

BUNDLEHASH

- Use the cryptographic sponge construction to absorb the bundleEssences and squeeze the hash.

```
bundle = [transactionObject0, transactionObject1, transactionObject2, transactionObject3]
transactionObjectN = {address, value, obsoleteTag, timestamp, currentIndex, lastIndex}
```



BUNDLEHASH

- Convert the hash to trytes:
  bundleHash = convertToTrytes(hash)

TRANSACTIONOBJECT EXAMPLE

- This is what a single transactionObject looks like in a transaction bundle:


```
{
    "hash": "YDDQ...A9999",
    "signatureMessageFragment": "JHAK...MVGY",
    "address": "HRKD...XKHX",
    "value": -3,
    "obsoleteTag": "999999999999999999999999999",
    "timestamp": 1515494426,
    "currentIndex": 1,
    "lastIndex": 2,
    "bundle": "RTGX...LQCY",  ⟵ bundleHash
    "trunkTransaction": "WVCLP...99999",
    "branchTransaction": "DOXV...X999",
    "tag": "999999999999999999999999999",
    "attachmentTimestamp": 1515496571334,
    "attachmentTimestampLowerBound": 0,
    "attachmentTimestampUpperBound": 3812798742493,
    "nonce": "AZ999IOB99999999999999999",
    "persistence": true
},
```

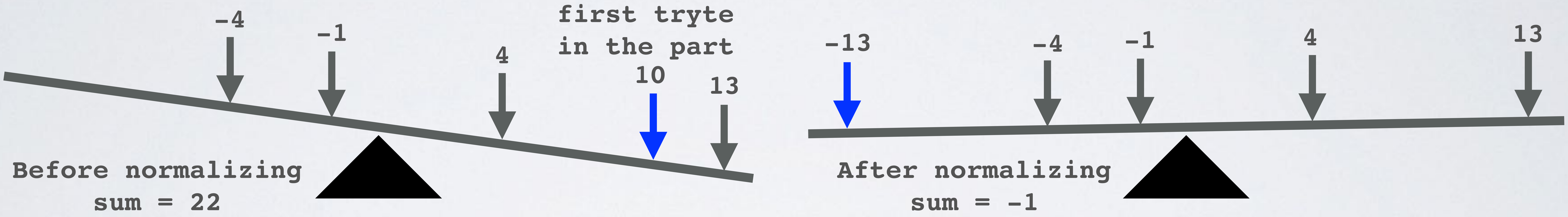
NORMALIZED BUNDLEHASH

- The normalizedBundleHash is created by extracting the bundleHash from the transactionObject and the bundleHash is then normalized.

- The normalizedBundleHash contains no tryte value M and the "weights" of the trytes are evenly distributed.

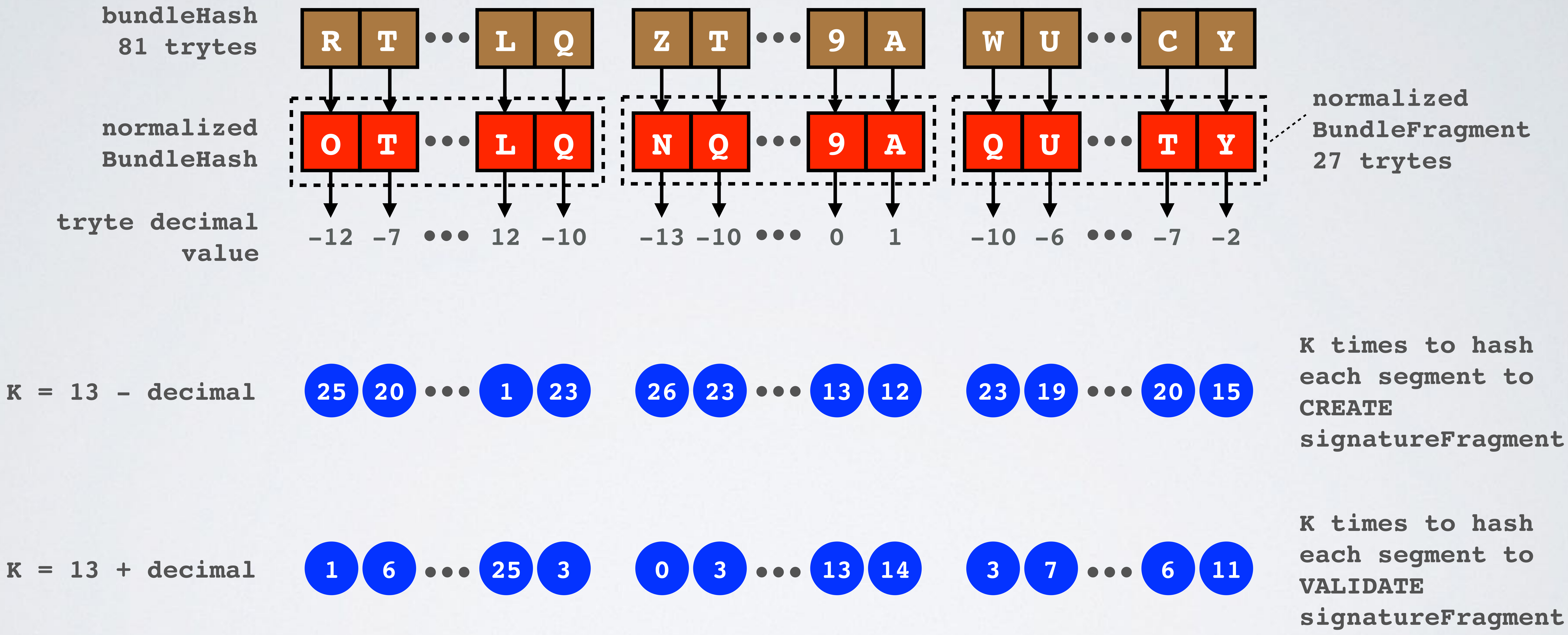
NORMALIZED BUNDLEHASH

- You can think of normalizing the bundleHash as balancing a seesaw, by manipulating its "weight" (=trytes) to reach a more equilibrium state.



- The normalizedBundleHash is used to create or validate a signature.

CALCULATE NUMBER OF HASHES

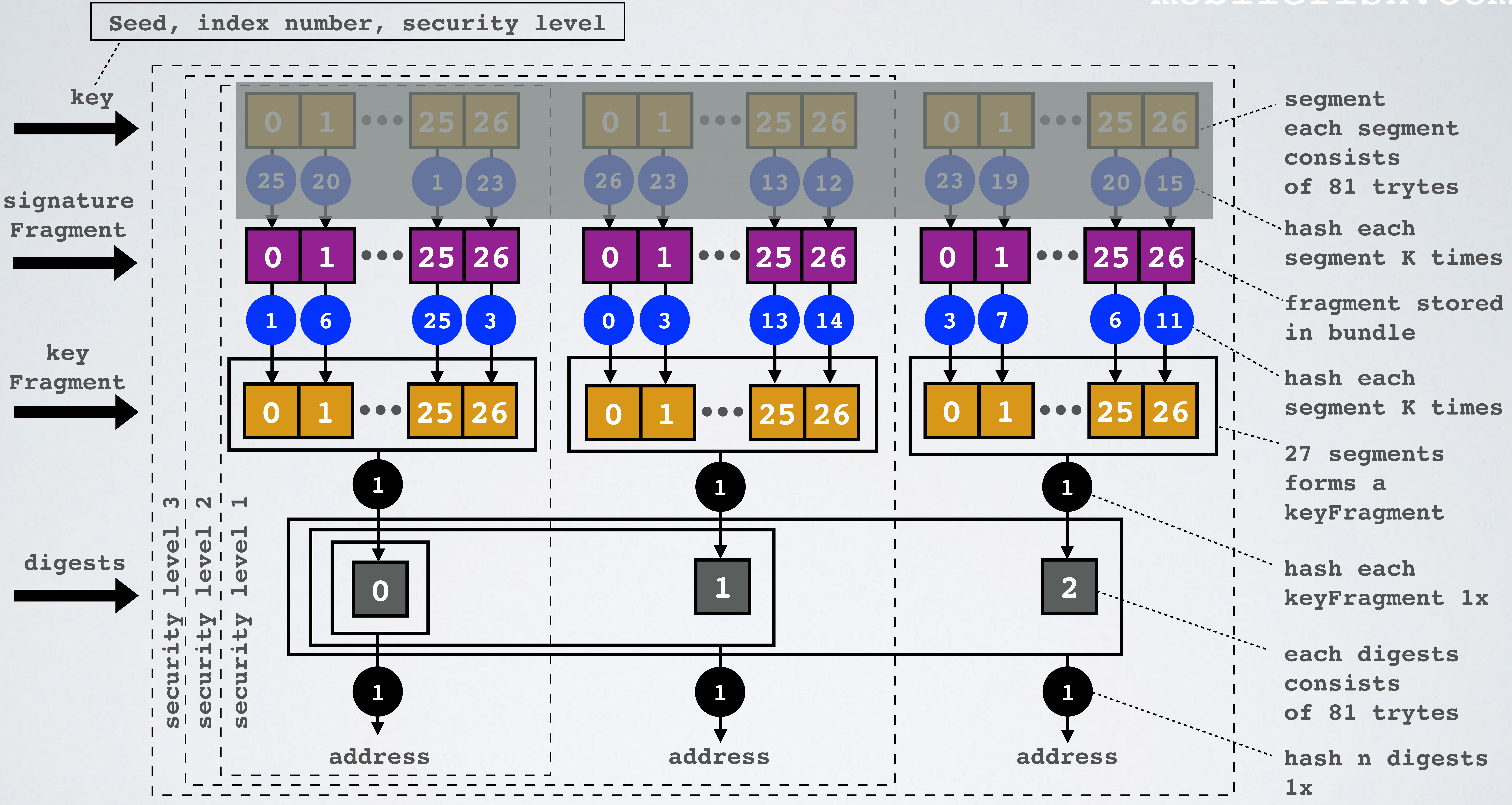


mobilefish.com
Seed, index number, security level
key
signature Fragment
key Fragment
digests
security level 3
security level 2
security level 1
0 1 ••• 25 26
25 20 1 23
26 23 13 12
23 19 20 15
1 6 25 3
0 3 13 14
3 7 6 11
0
1
2
address
address
address
segment
each segment consists of 81 trytes
hash each segment K times
fragment stored in bundle
hash each segment K times
27 segments forms a keyFragment
hash each keyFragment 1x
each digests consists of 81 trytes
hash n digests 1x

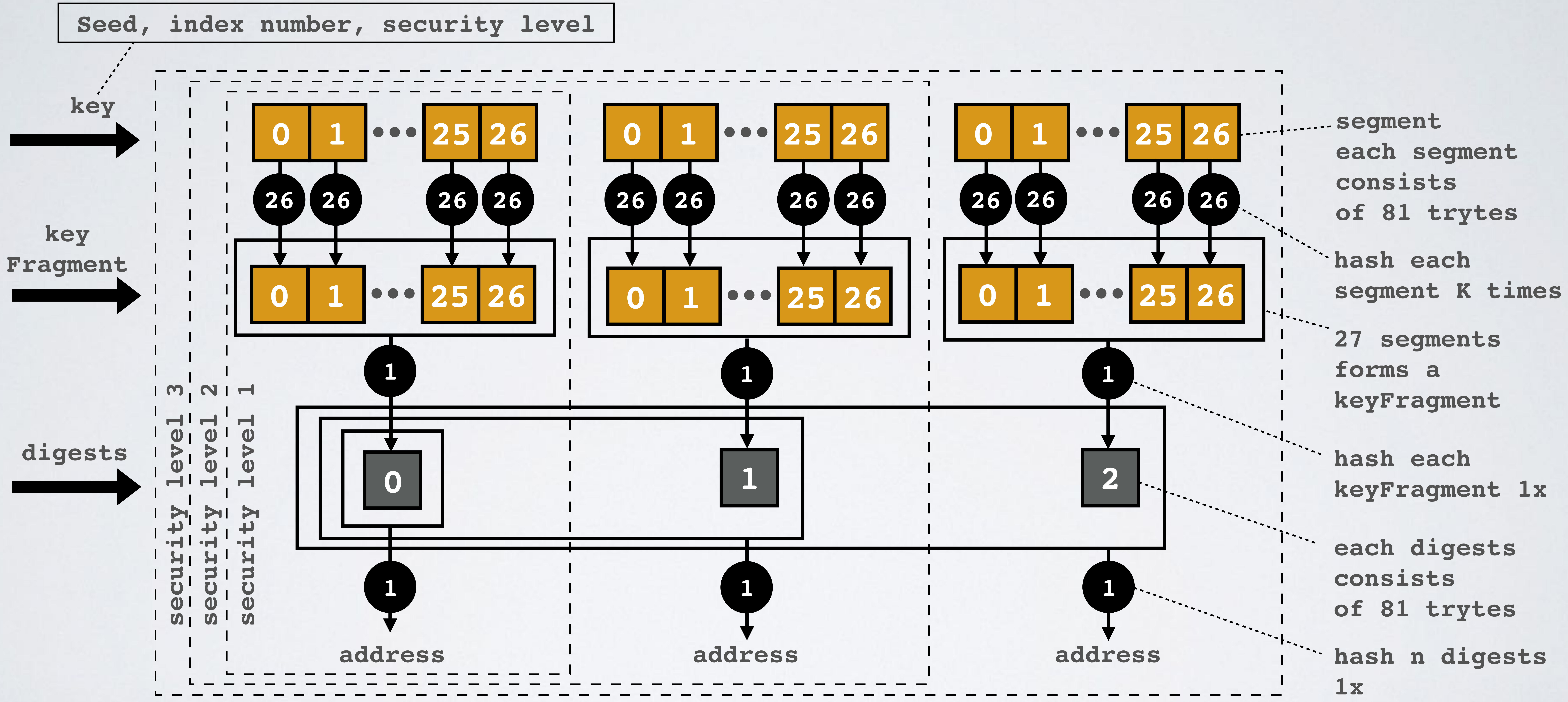
SIGNATUREMESSAGEFRAGMENT EXAMPLE

- This is what a single transactionObject looks like in a transaction bundle:


```
{
    "hash": "YDDQ...A9999",
    "signatureMessageFragment": "JHAK...MVGY",
    "address": "HRKD...XKHX",
    "value": -3,
    "obsoleteTag": "999999999999999999999999999",
    "timestamp": 1515494426,
    "currentIndex": 1,
    "lastIndex": 2,
    "bundle": "RTGX...LQCY",
    "trunkTransaction": "WVCLP...99999",
    "branchTransaction": "DOXV...X999",
    "tag": "999999999999999999999999999",
    "attachmentTimestamp": 1515496571334,
    "attachmentTimestampLowerBound": 0,
    "attachmentTimestampUpperBound": 3812798742493,
    "nonce": "AZ999IOB999999999999999999",
    "persistence": true
},
```


Seed, index number, security level
key
signature Fragment
key Fragment
digests
security level 3
security level 2
security level 1
0 1 ••• 25 26
25 20 1 23
26 23 13 12
23 19 20 15
1 6 25 3
0 3 13 14
3 7 6 11
0 1 2
address
address
address
segment
each segment consists of 81 trytes
hash each segment K times
fragment stored in bundle
hash each segment K times
27 segments forms a keyFragment
hash each keyFragment 1x
each digests consists of 81 trytes
hash n digests 1x

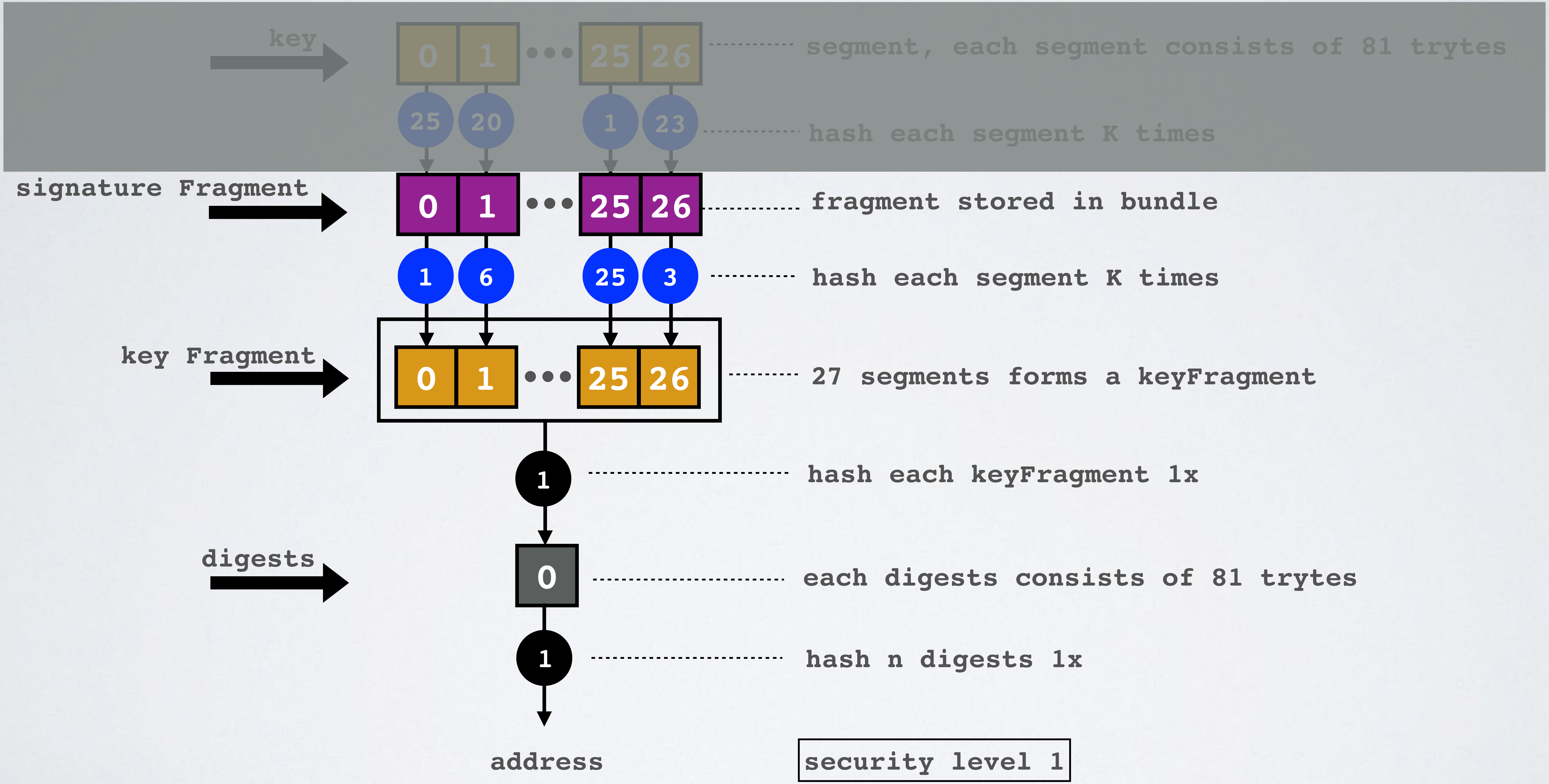
CALCULATE ADDRESS



WHY IS NORMALIZEDBUNDLEHASH NEEDED?

- I have created a simple value transaction: I have used security level 1 and transferred 1 IOTA from address A to B and there is no remainder.
- See the corresponding transaction bundle:
  https://www.mobilefish.com/download/iota/transactions_in_bundle_security_level1.txt
- The transaction bundle has two transactionObjects.
  A transactionObject containing recipient data and the other containing sender data.
- The senders signatureMessageFragment is "KVSA…HMKW" and the senders address is "VXO…LTKA".

WHY IS NORMALIZEDBUNDLEHASH NEEDED?



WHY IS NORMALIZEDBUNDLEHASH NEEDED?

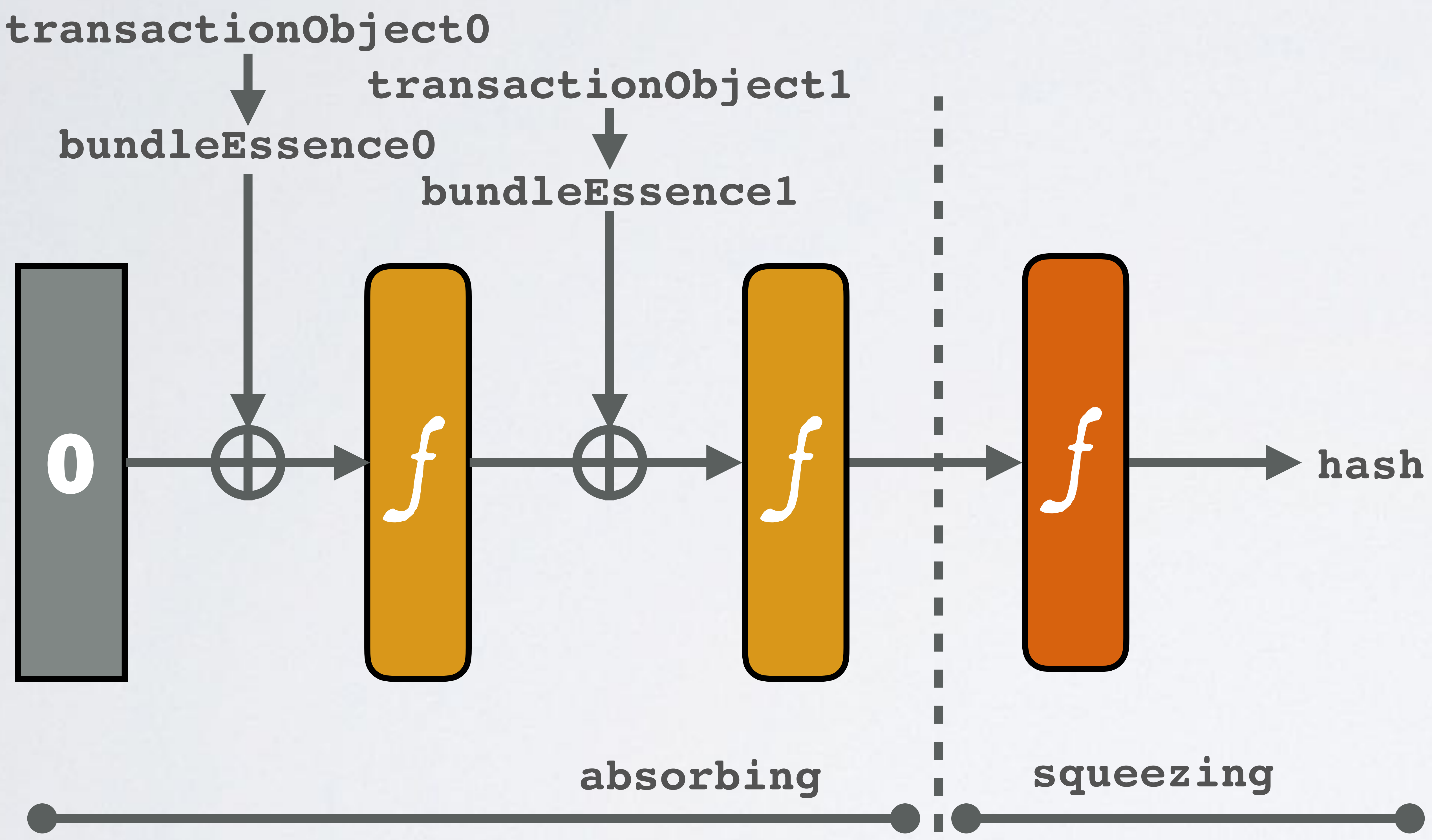
- Let assume the submitted transaction bundle is pending and a hacker, called Eve, gets hold of this transaction bundle.

- Eve can change the transaction bundle by replacing the recipient's address with her own address. By doing so the bundleHash changes which means the normalizedBundleHash and the number of hashes (K) are also changed accordingly.

WHY IS NORMALIZEDBUNDLEHASH NEEDED?

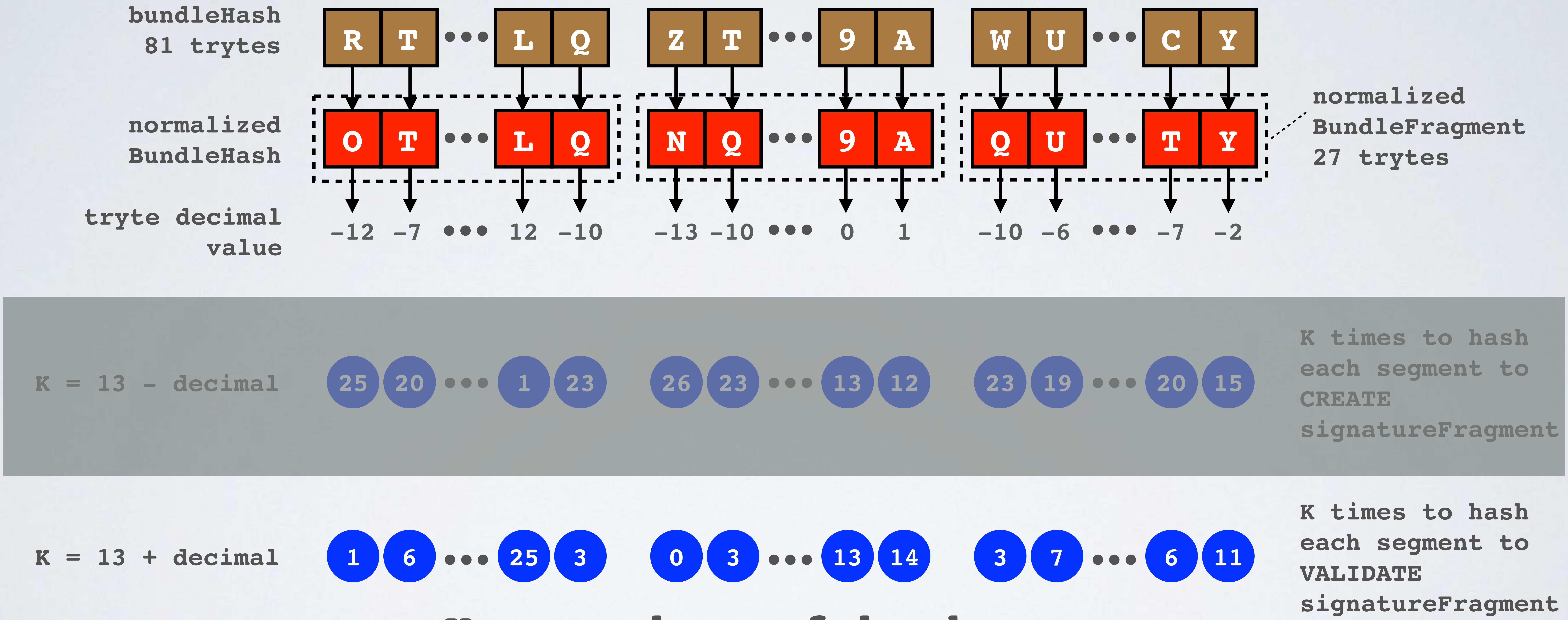
```
bundle = [transactionObject0, transactionObject1]

transactionObjectN = {address, value, obsoleteTag, timestamp, currentIndex, lastIndex}
```



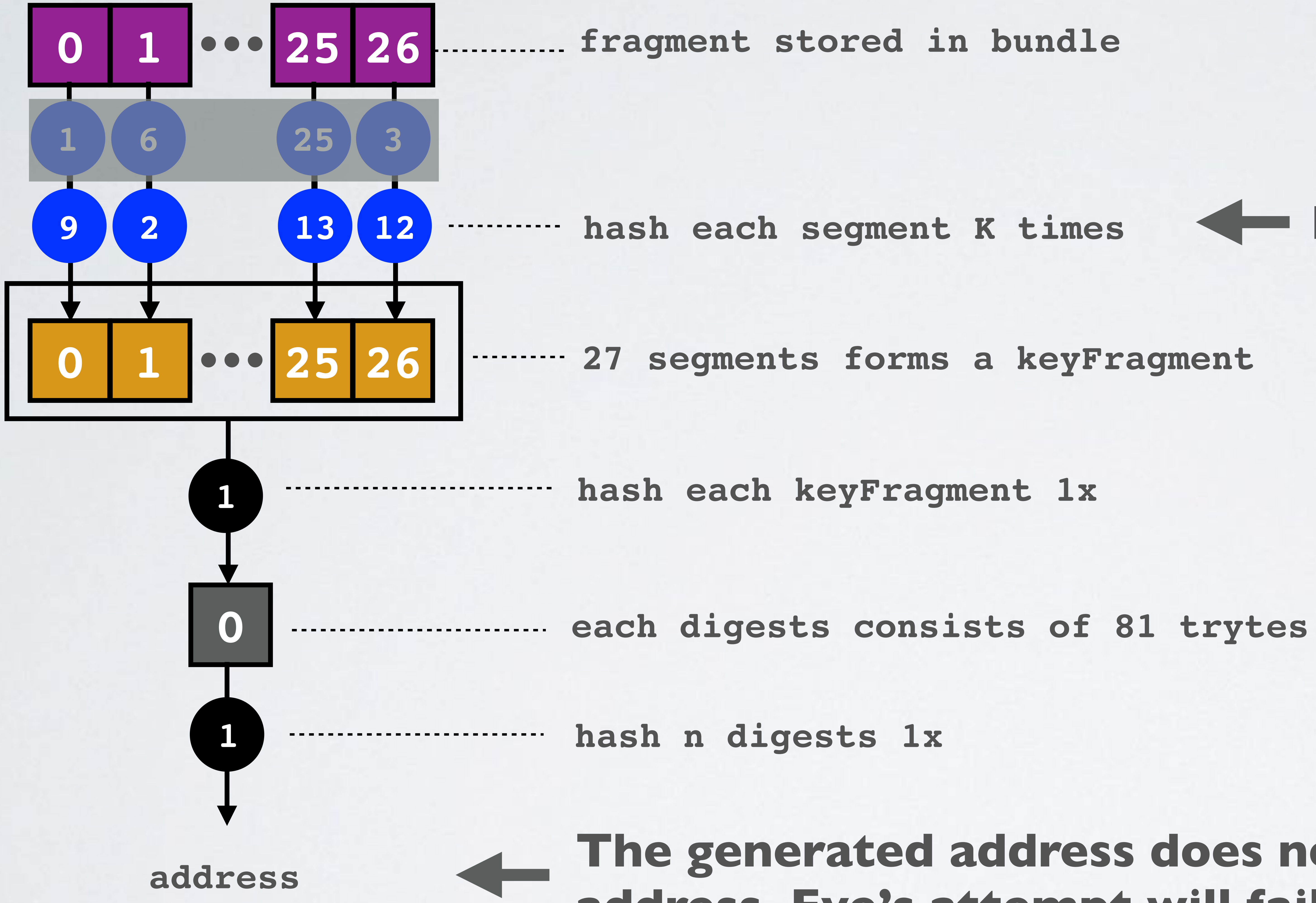

```
bundleHash = convertToTrytes(hash)
```

WHY IS NORMALIZEDBUNDLEHASH NEEDED?



K = number of hashes

WHY IS NORMALIZEDBUNDLEHASH NEEDED?



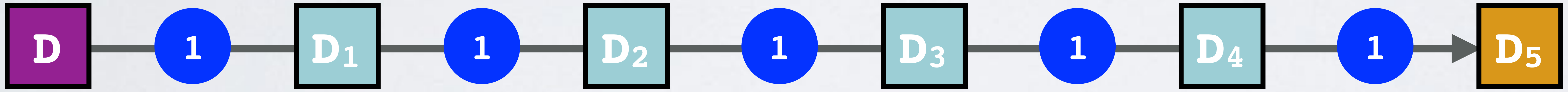
Number of hashes (K) changed!

The generated address does not match the senders address. Eve's attempt will fail.

WHY IS NORMALIZEDBUNDLEHASH NEEDED?

Data (D) is hashed 5x to get the hashed result D_5:

You can also draw it this way:



Question: Can you hash a value 3x to get D_5?
Answer: Yes, if you start with D_2.

Question: Can you hash a value 6x to get D_5?
Answer: No, you can't! A hash algorithm is a one-way function.

WHY IS NORMALIZEDBUNDLEHASH NEEDED?

Data (D) is hashed 5x to get the hashed result D_5:

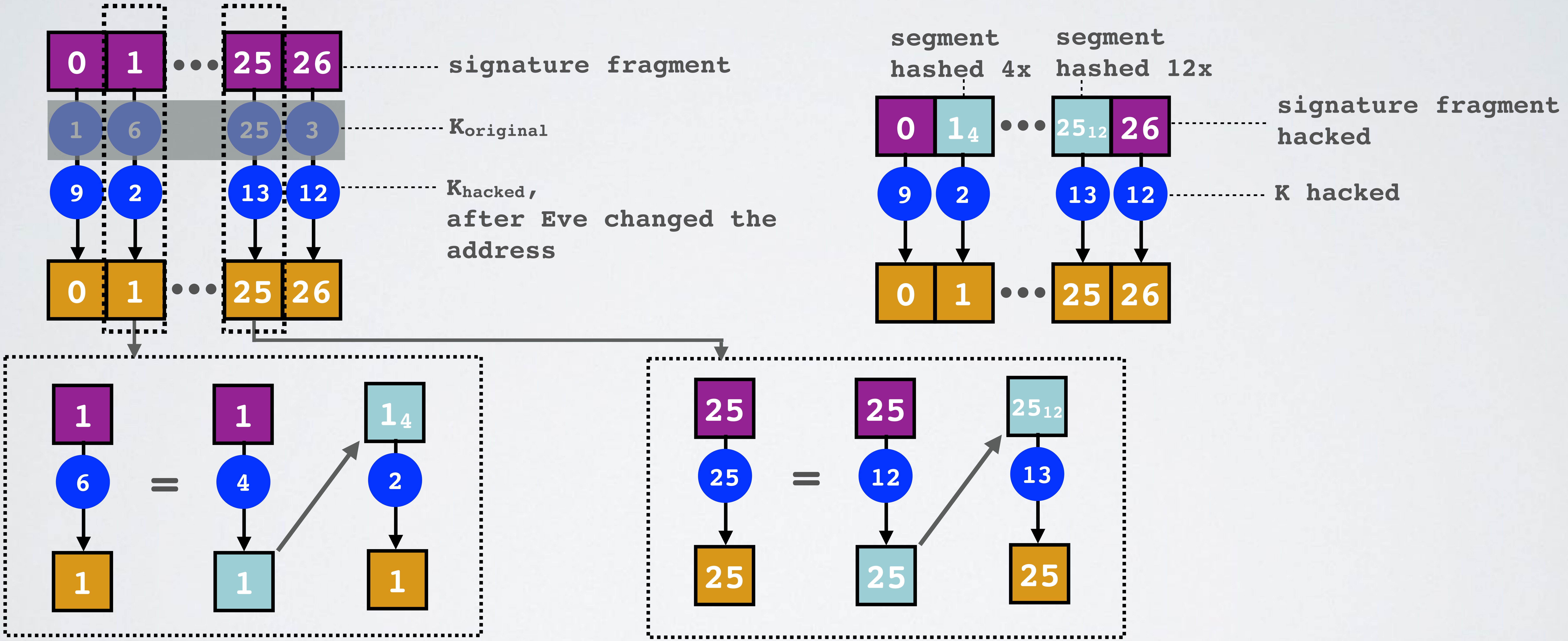
Question: Can you hash a value 1x to get D_5?
Answer: Yes, if you start with D_4.

Question: Can you hash a value 4x to get D_5?
Answer: Yes, if you start with D_1.

Question: Can you hash a value 7x to get D_5?
Answer: No, you can't.

WHY IS NORMALIZEDBUNDLEHASH NEEDED?

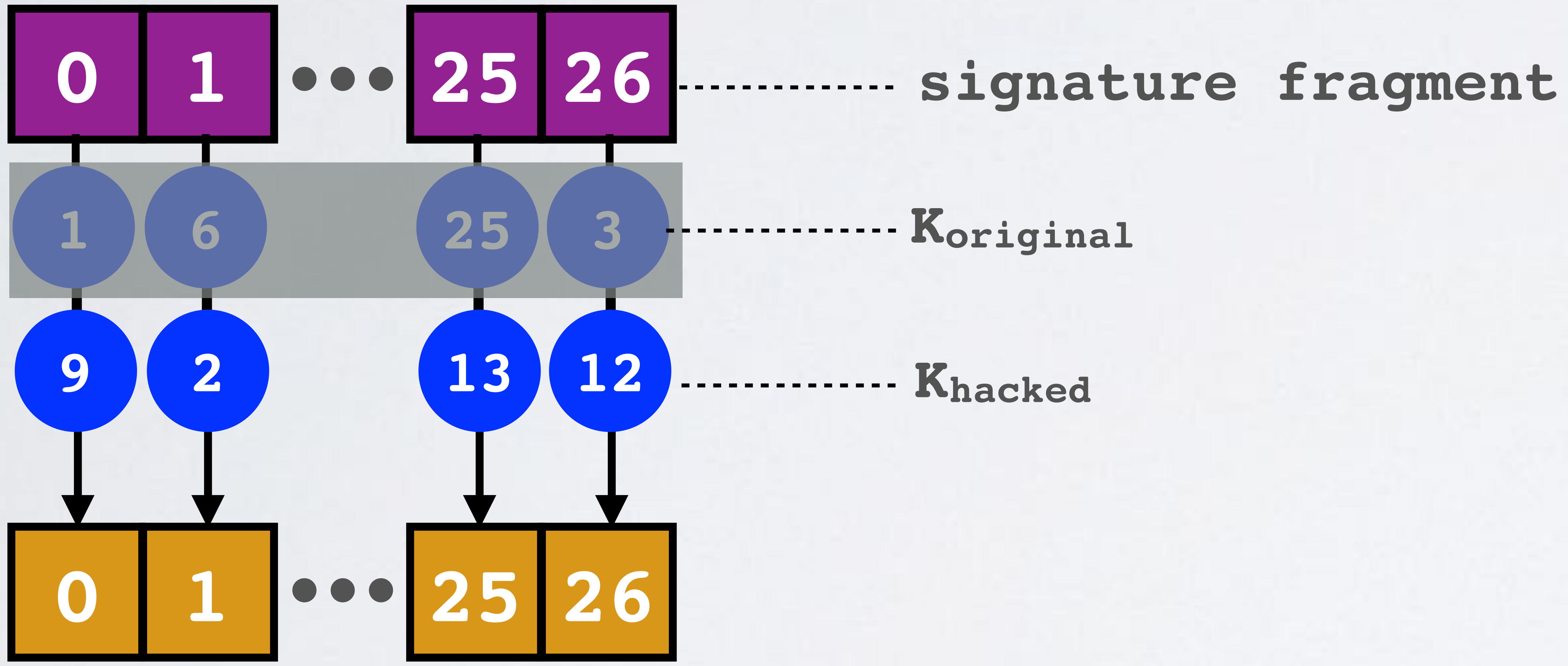
- Eve knows she can change the signatureFragment.

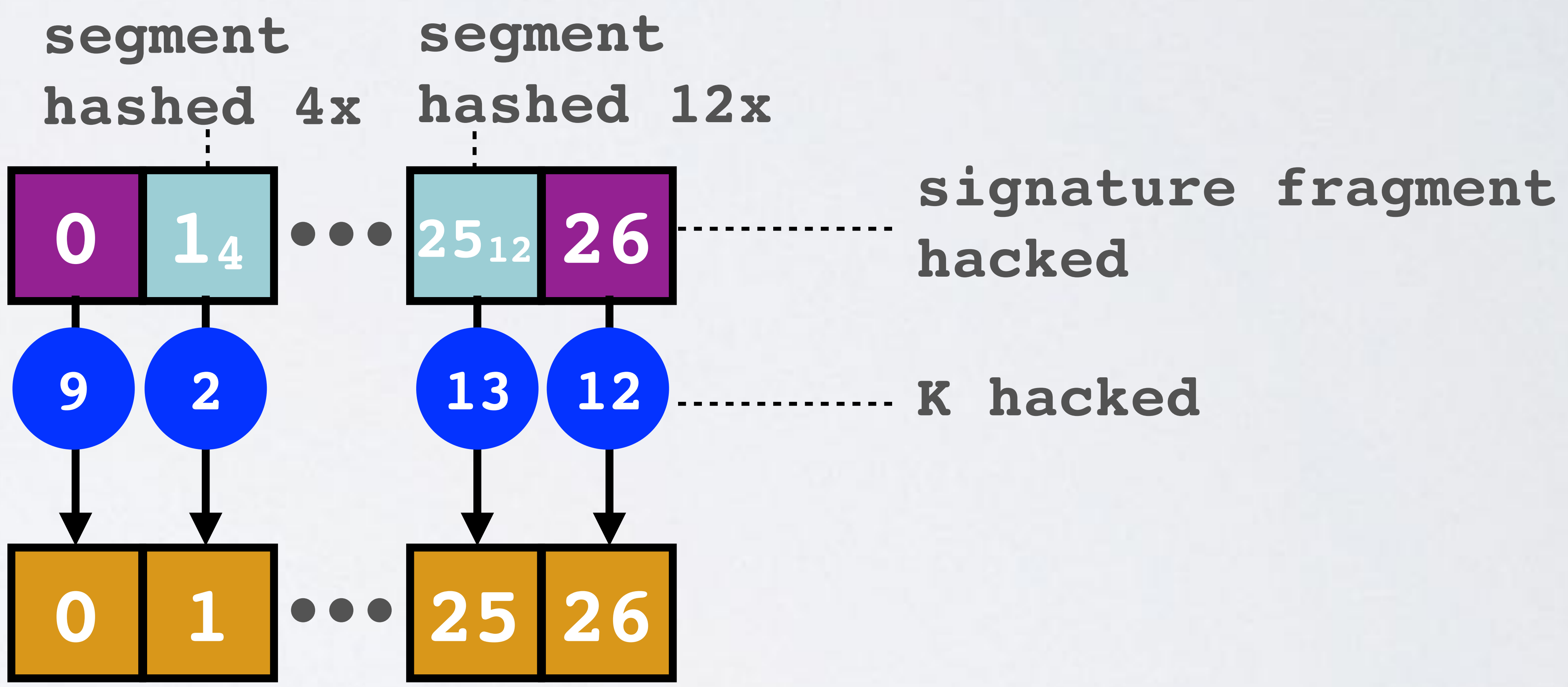


mobilefish.com

WHY IS NORMALIZEDBUNDLEHASH NEEDED?

- But Eve still has a problem with the first and last segment.
- Her attempt is only successful if all $K_{original}$ values are bigger of equal than the corresponding K_{hacked} values.

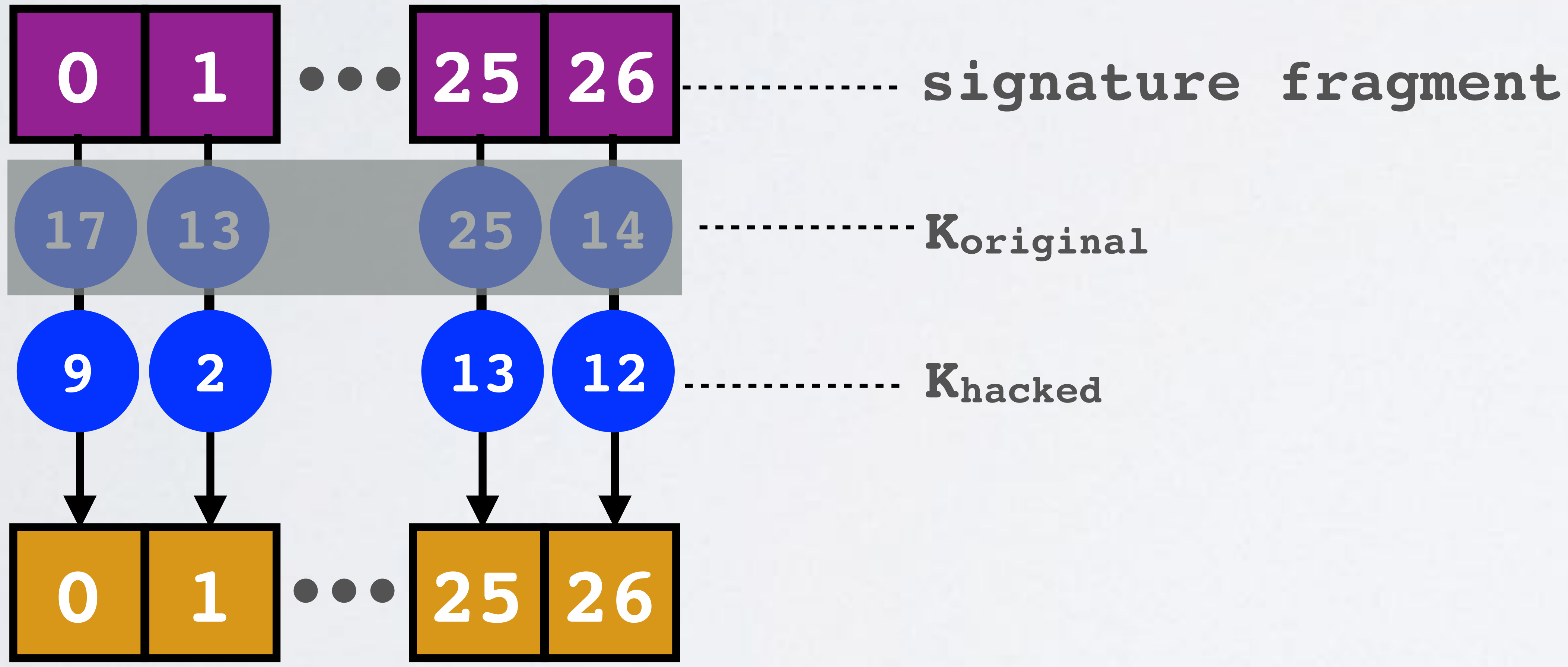


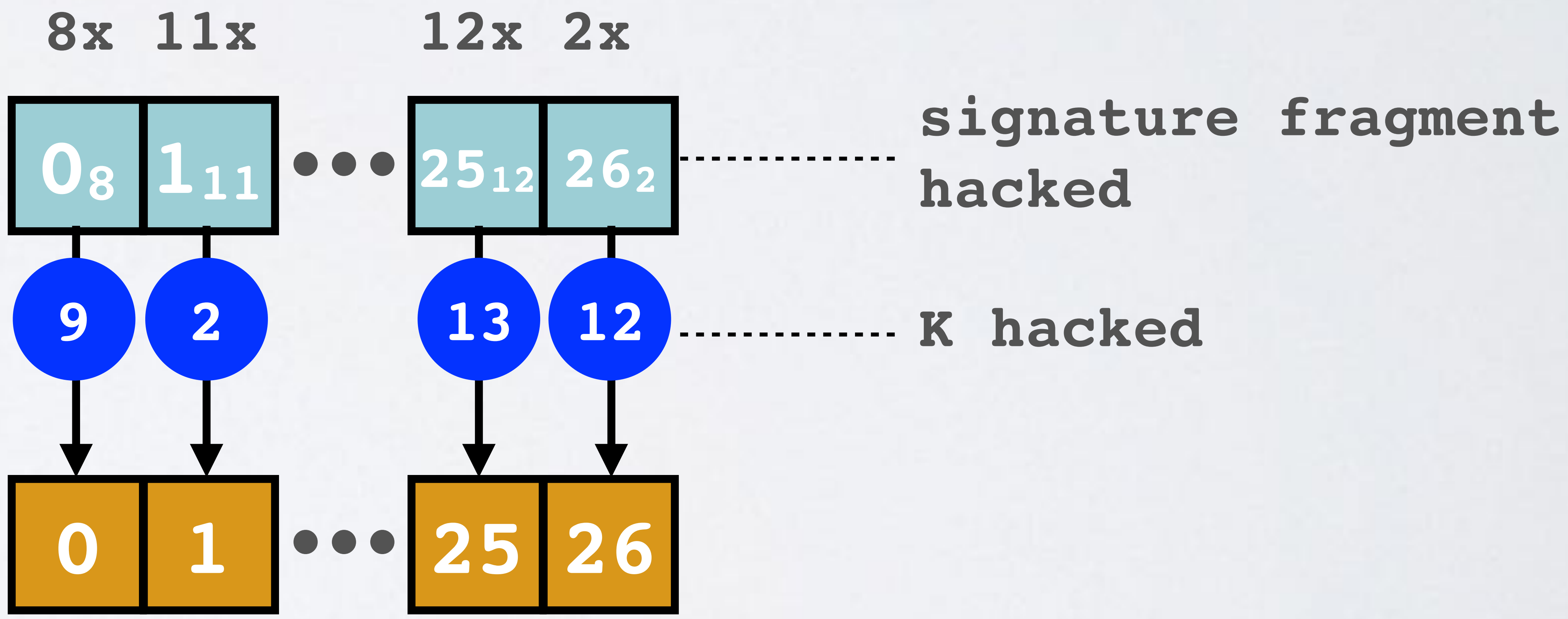


WHY IS NORMALIZEDBUNDLEHASH NEEDED?

- Now lets assume the following case:
  The $K_{original}$ values are all between 14-26.
  The K_{hacked} values are all between 1-13.

- In this case, Eve can successfully hack the transaction bundle and send IOTAs to her address.

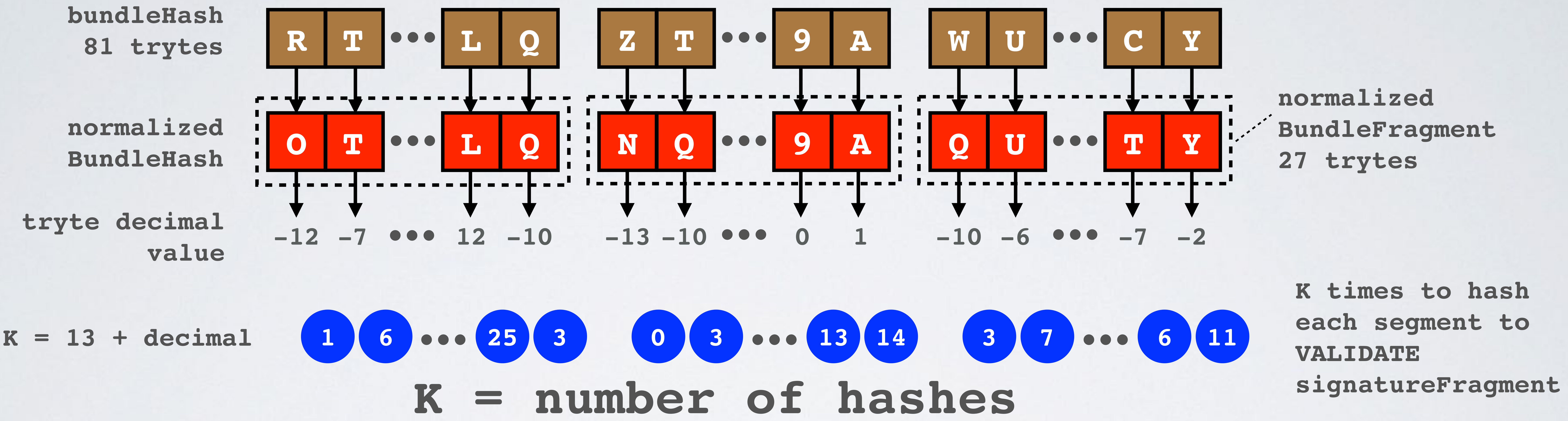




WHY IS NORMALIZEDBUNDLEHASH NEEDED?

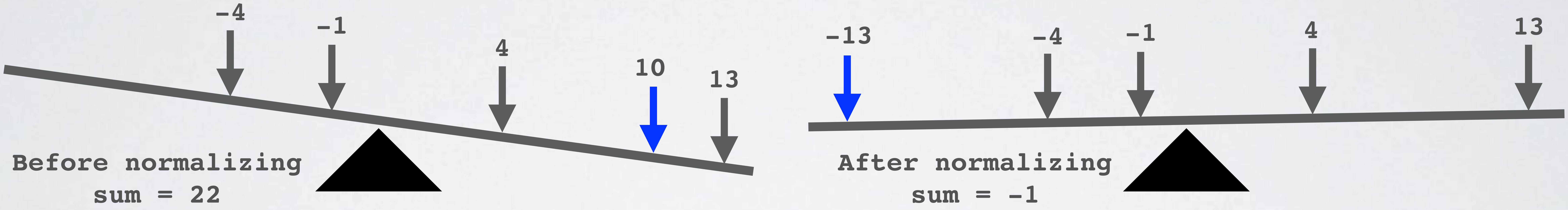
- However in reality the previous mentioned case is difficult to realise because a normalizedBundleHash is used.
- Eve attempt can only be successful if ALL $K_{original}$ values are bigger of equal than the corresponding K_{hacked} values.
- By using a normalizedBundleHash the probability that this will happen is small.

WHY IS NORMALIZEDBUNDLEHASH NEEDED?



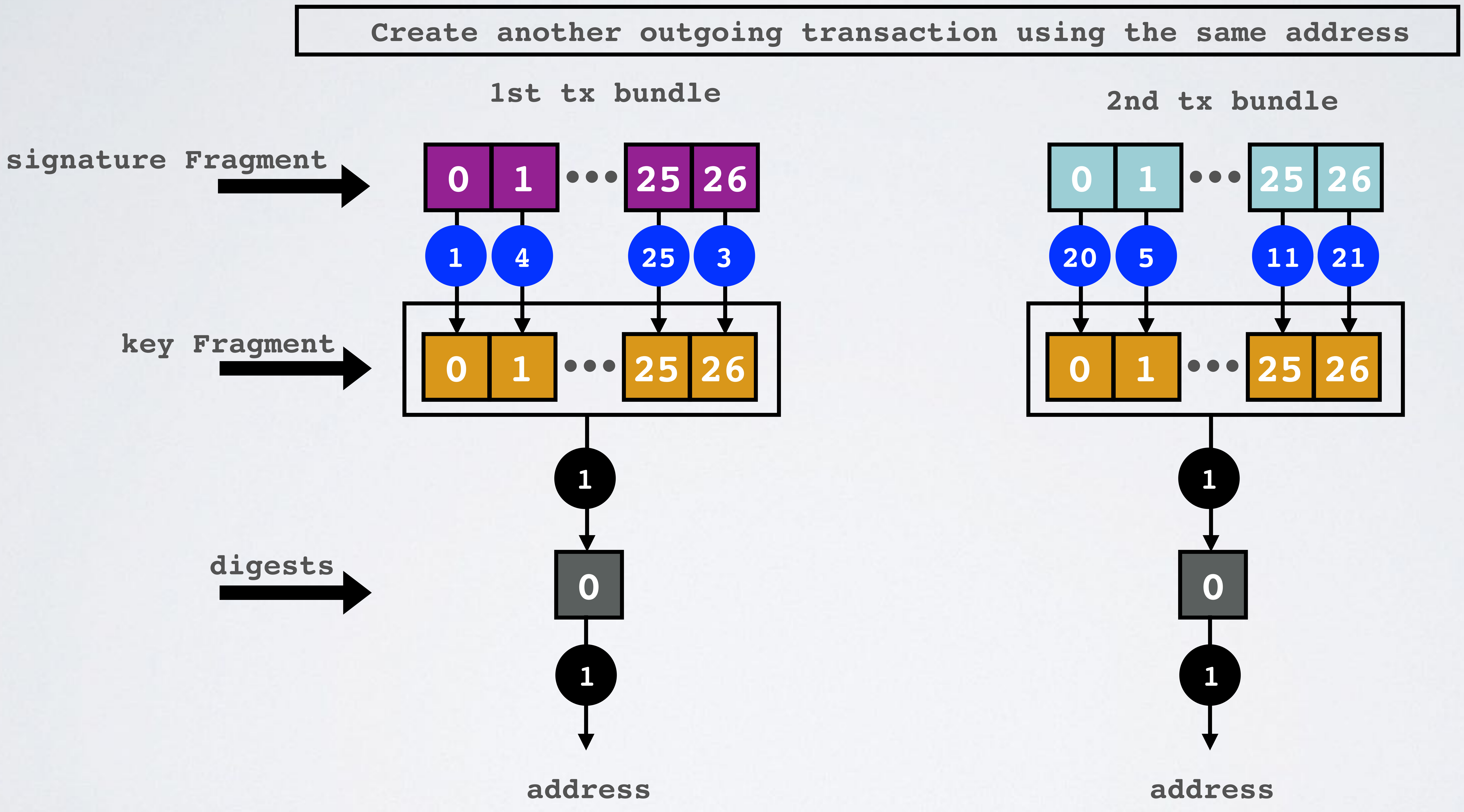
WHY IS NORMALIZEDBUNDLEHASH NEEDED?

- The previous mentioned decimal values (= normalizedBundleHash tryte values converted to decimal values) are in the range -13 to 13 and are evenly distributed just like a seesaw.



- By distributing these values evenly the $K_{original}$ values are "spread".
  You will have low values: 1-13 and high values 14-26.
  You can not have only $K_{original}$ values between 14 and 26, the normalizedBundleHash prevents this.

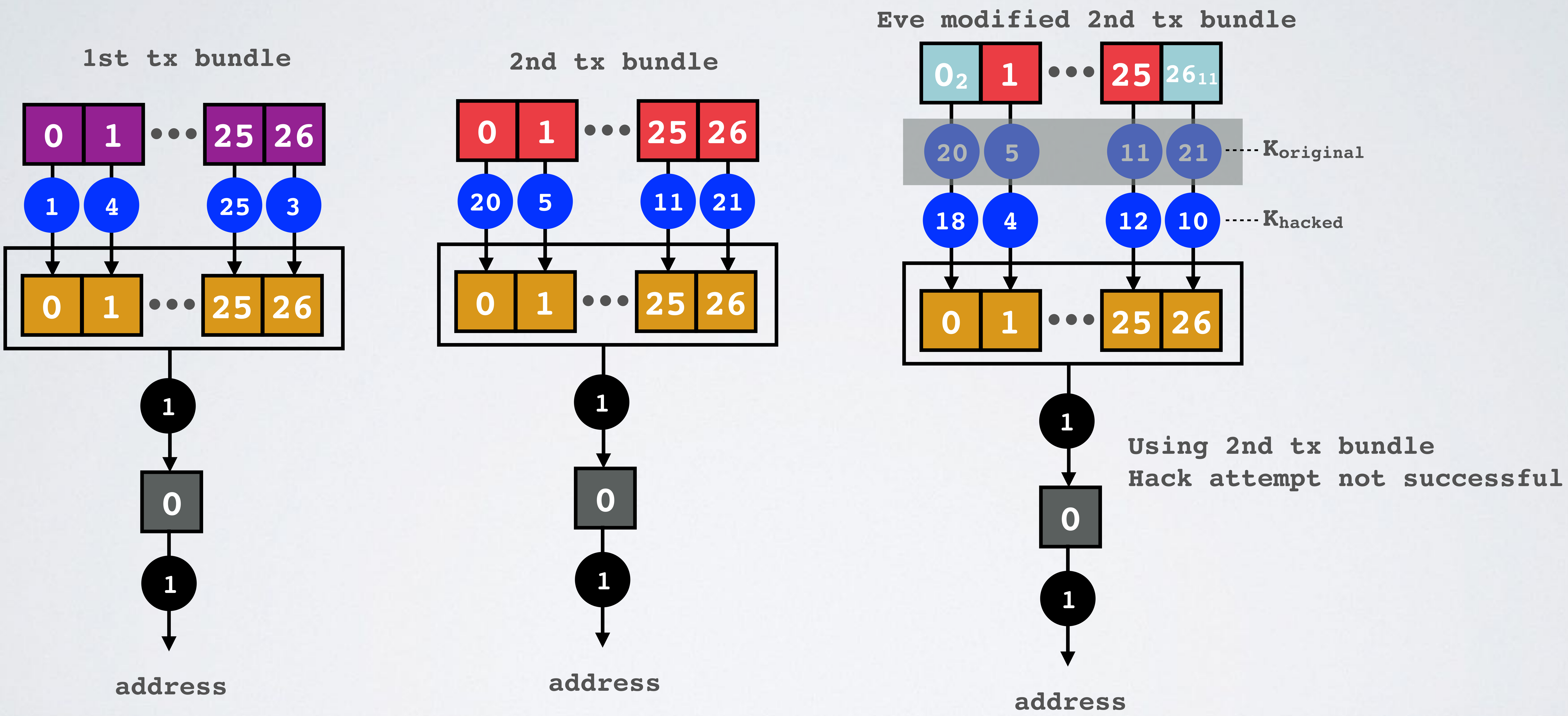
WHY NOT REUSE AN ADDRESS FOR OUTGOING TXS?



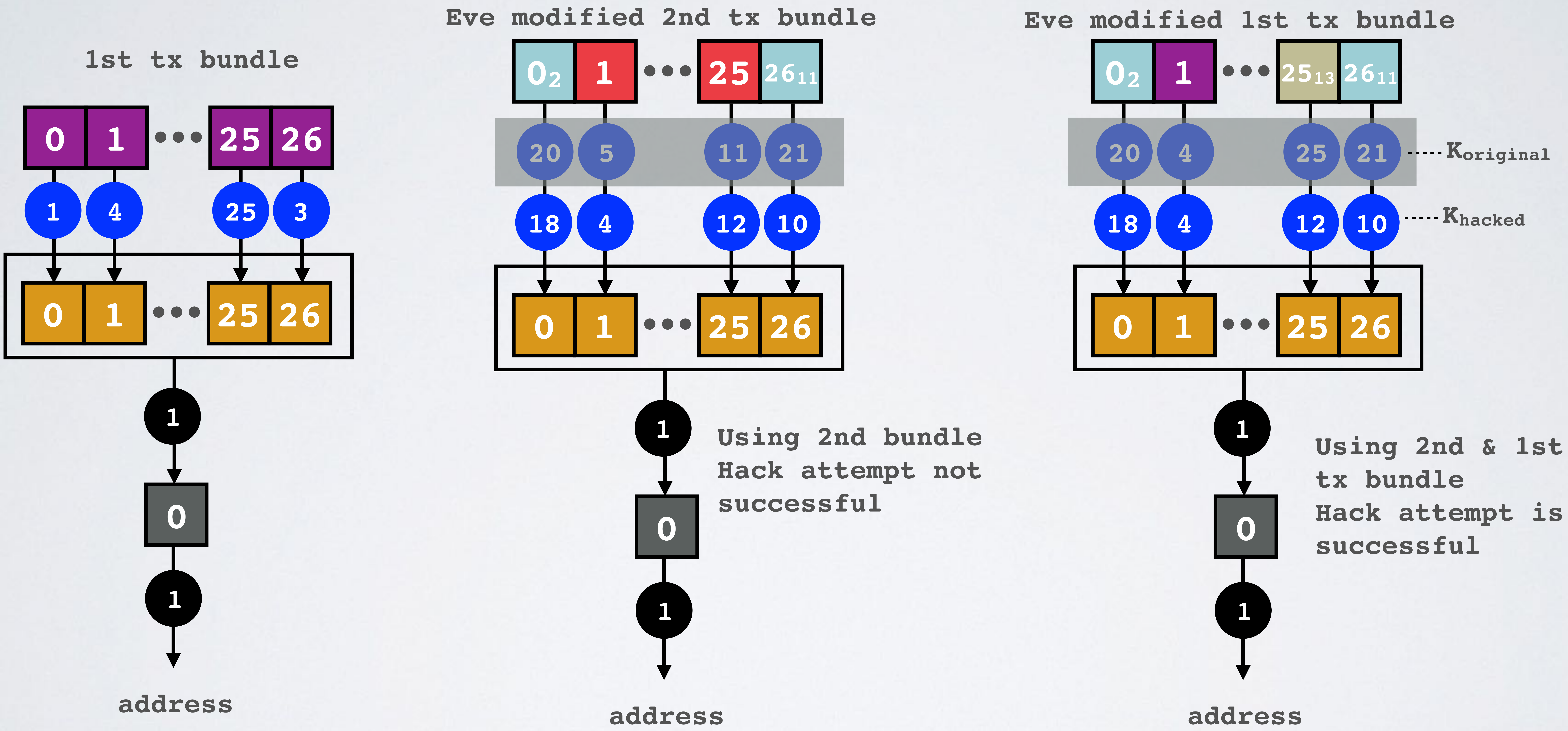
WHY NOT REUSE AN ADDRESS FOR OUTGOING TXS?

- Eve has found these two transaction bundles using the same address A for outgoing transactions.
- A few days later, Eve noticed 500 MIOTA were send to address A.
- Eve tries a hack attempt, she takes the 2nd transaction bundle:
    - From the receiver tx object, she change the recipient's address with her own address and change the recipient's value to 500 MIOTA.
    - From the sender tx object, she change the spending value to 500 MIOTA.
- By doing so the bundleHash, normalizedBundleHash and the K values are changed.

WHY NOT REUSE AN ADDRESS FOR OUTGOING TXS?



WHY NOT REUSE AN ADDRESS FOR OUTGOING TXS?



WHY NOT REUSE AN ADDRESS FOR OUTGOING TXS?

- If you reuse an address for outgoing addresses you provide a hacker more possibilities to successfully create a modified transaction bundle sending IOTAs from the victim's address to the hackers address.

- Reusing an address for outgoing transactions does not mean the hacker will immediately succeed in its hack attempt, but it will definitely increase its chances.