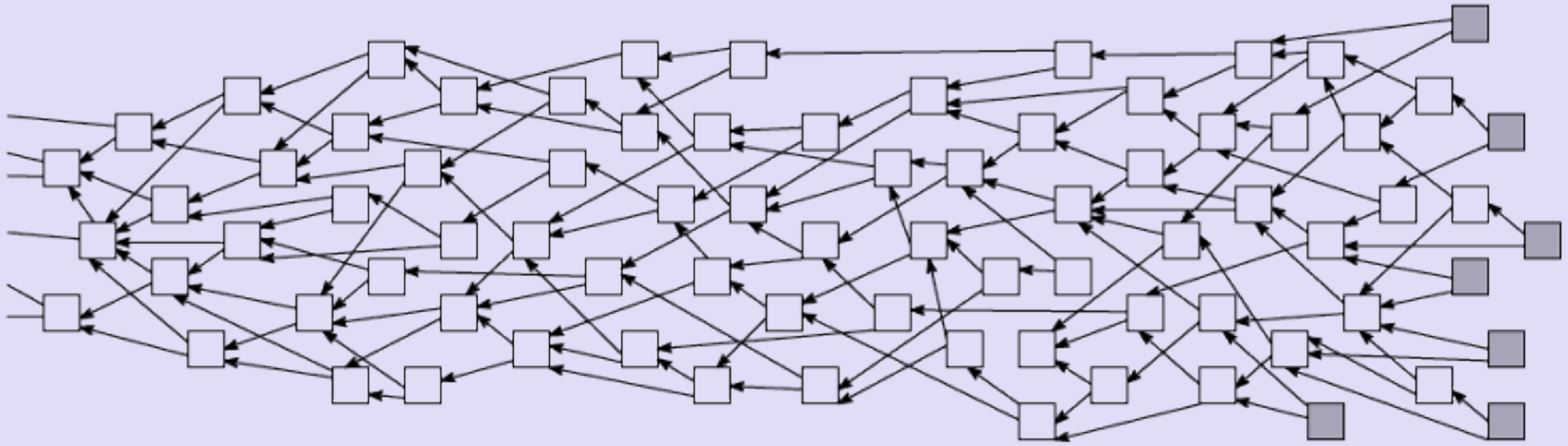
IOTA TUTORIAL 24

Multisignatures



G. W. Carver

Abraham Lincoln

INTRO

- In this video I will explain what the purpose is of multisignatures and how to use IOTA multisignatures.

MULTISIGNATURES

- Multisignature (multisig) refers to requiring multiple signatures to authorise a transaction.

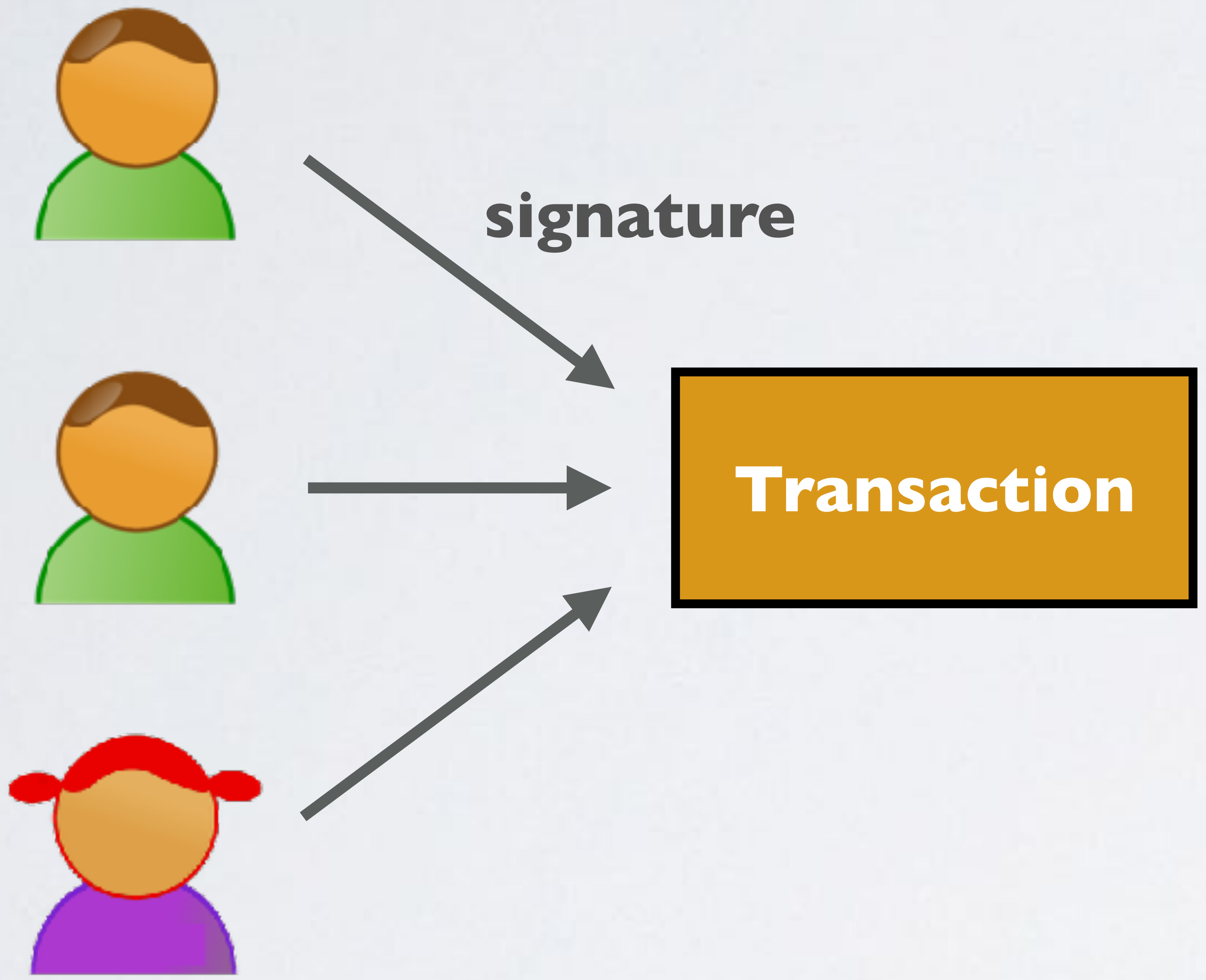
- There are two types of multisignature schemes:

  - **N-of-N scheme**
    All co-signers (N) needs to provide their signatures in order for a transfer to be successful.

  - **M-of-N scheme**
    M co-signers of a total of N co-signers needs to provide their signatures in order for a transfer to be successful.

MULTISIGNATURE SCHEME EXAMPLE 1

N-of-N scheme

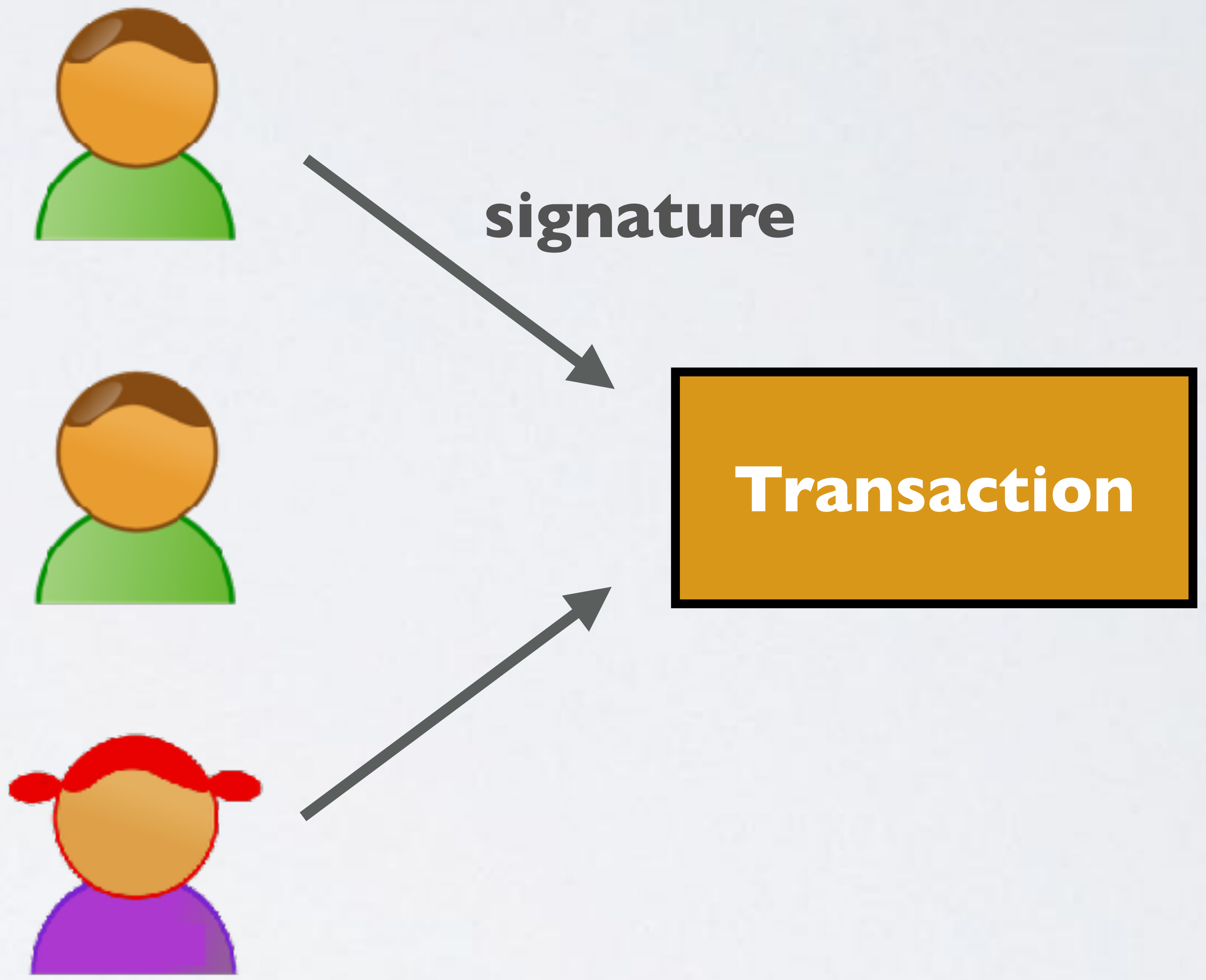


signature

Transaction

3-of-3

M-of-N scheme



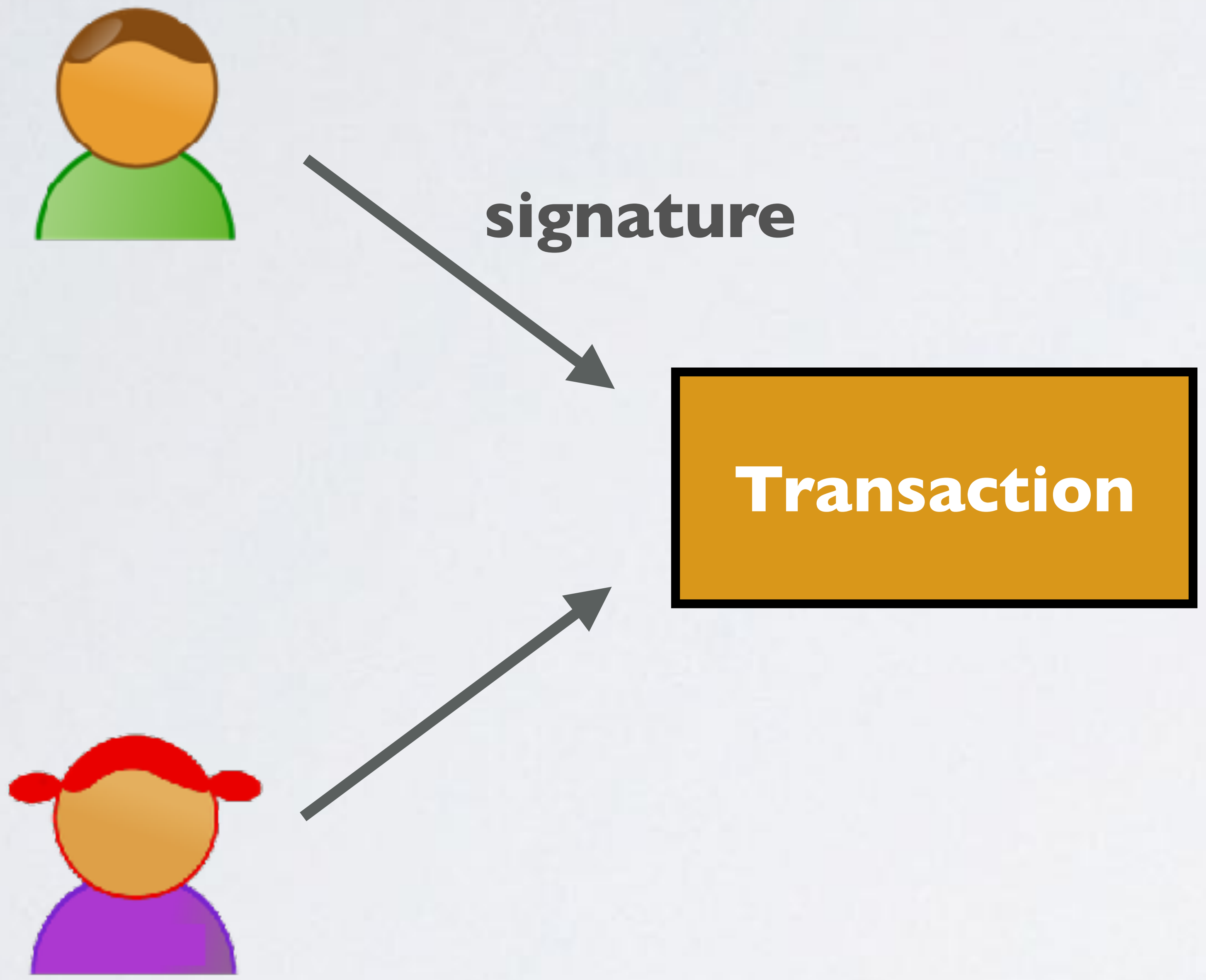
signature

Transaction

2-of-3

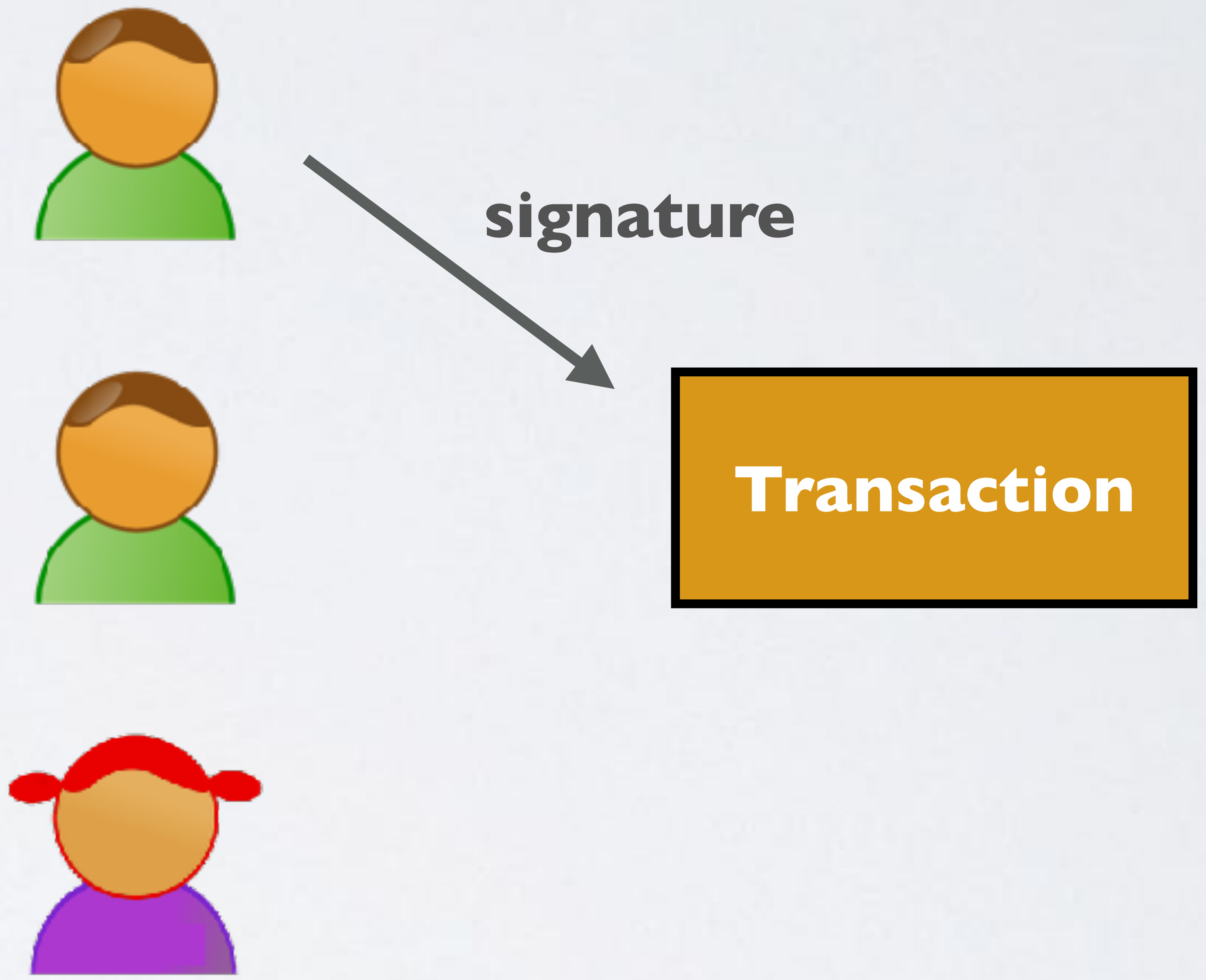
MULTISIGNATURE SCHEME EXAMPLE 2

N-of-N scheme



2-of-2

M-of-N scheme



1-of-3

N-OF-N SCHEME

- In this tutorial lets assume Alice is the notary, Bob and Charlie are business companions buying an office space worth 2 IOTAs. (Yes, that is cheap!)

- Alice uses a 3-of-3 scheme, which means 3 signatures (Alice, Bob and Charlie) are required for a successful transaction.

- Alice is the organiser and will handle the multisignature process.

- For educational purpose you can use the Multisig Wallet: https://www.mobilefish.com/services/cryptocurrency/iota_multisig.html

- **DO NOT USE THIS MULTISIG WALLET IN PRODUCTION YOU WILL LOSE ALL YOUR IOTAS!**

MULTISIGNATURE PROCESS: DIGEST

- Alice, Bob and Charlie each creates a digest using their seed and an unused key index. They can all use a different security level.

```
Alice:   seed=FTY…U9P, key index=24, security level=2
Bob:     seed=RRG…QWT, key index=0,  security level=1
Charlie: seed=XBH…EDF, key index=1,  security level=3
```

- The sum of the security levels (sumSecurityLevels) = 2 + 1 + 3 = 6
  This value will be used later.

- https://www.mobilefish.com/download/iota/multisig_demo.txt

MULTISIGNATURE PROCESS: DIGEST

- All multisig API's can be found in the IOTA Javascript library:
  https://github.com/iotaledger/iota.lib.js

- API to calculate the digest:
  digest = iota.multisig.getDigest(seed, keyIndex, securityLevel)

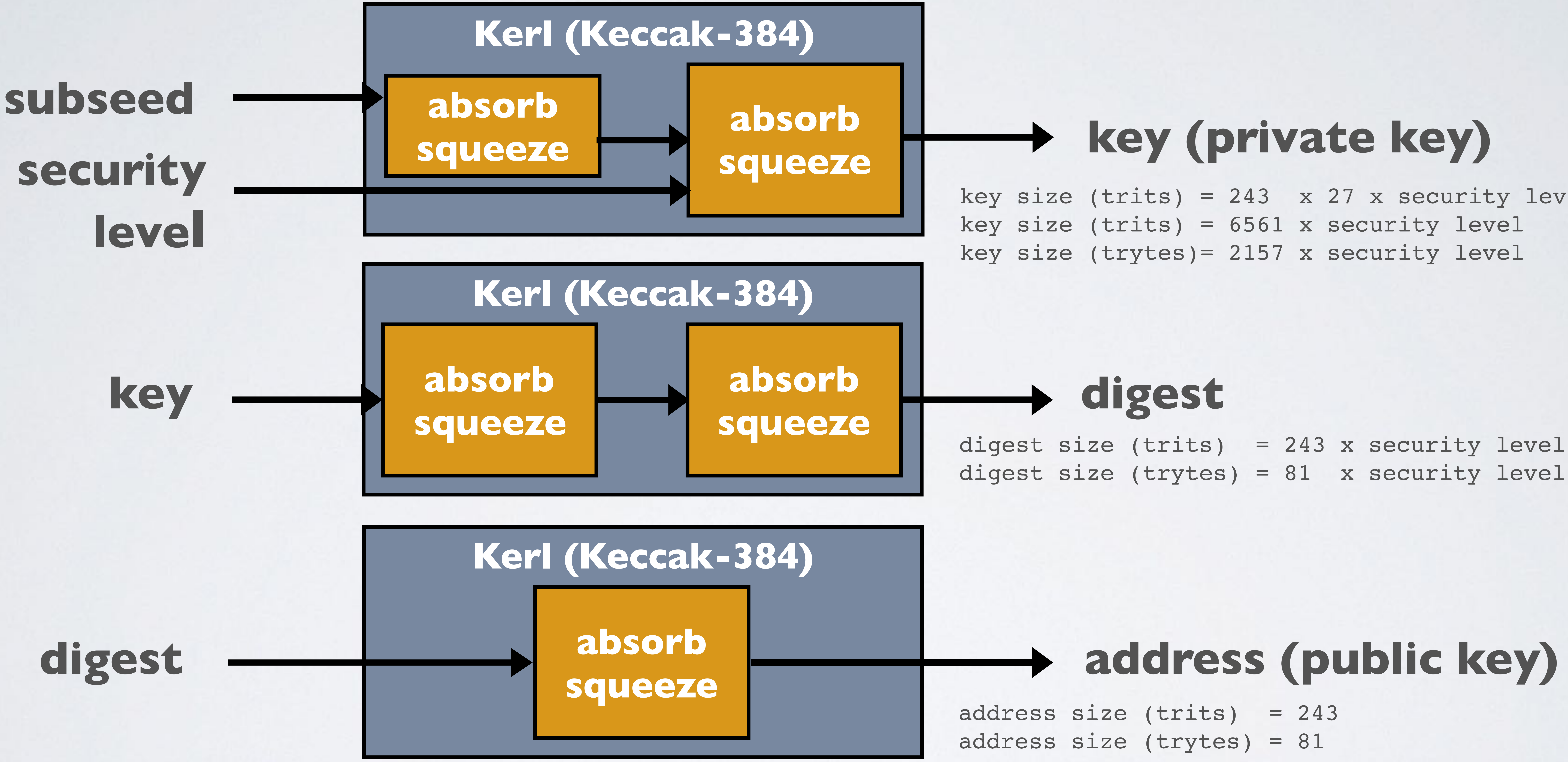
- The getDigest is just a wrapper, under the hood it calculates the subseed than the key and finally the digest.

CALCULATE SUBSEED

- Create a subseed by adding the seed and key index together.
  subseed = seed + key index

```
seed:       0,1,0,0,0,0 ... -1,-1,0,-1,1,-1
key index:  1,0,0
            ------------------------------- +
subseed:    1,1,0,0,0,0 ... -1,-1,0,-1,1,-1
```

CALCULATE DIGEST



subseed

security level

Kerl (Keccak-384)

absorb squeeze

absorb squeeze

key (private key)

key size (trits) = 243 x 27 x security level
key size (trits) = 6561 x security level
key size (trytes)= 2157 x security level

key

Kerl (Keccak-384)

absorb squeeze

absorb squeeze

digest

digest size (trits) = 243 x security level
digest size (trytes) = 81 x security level

digest

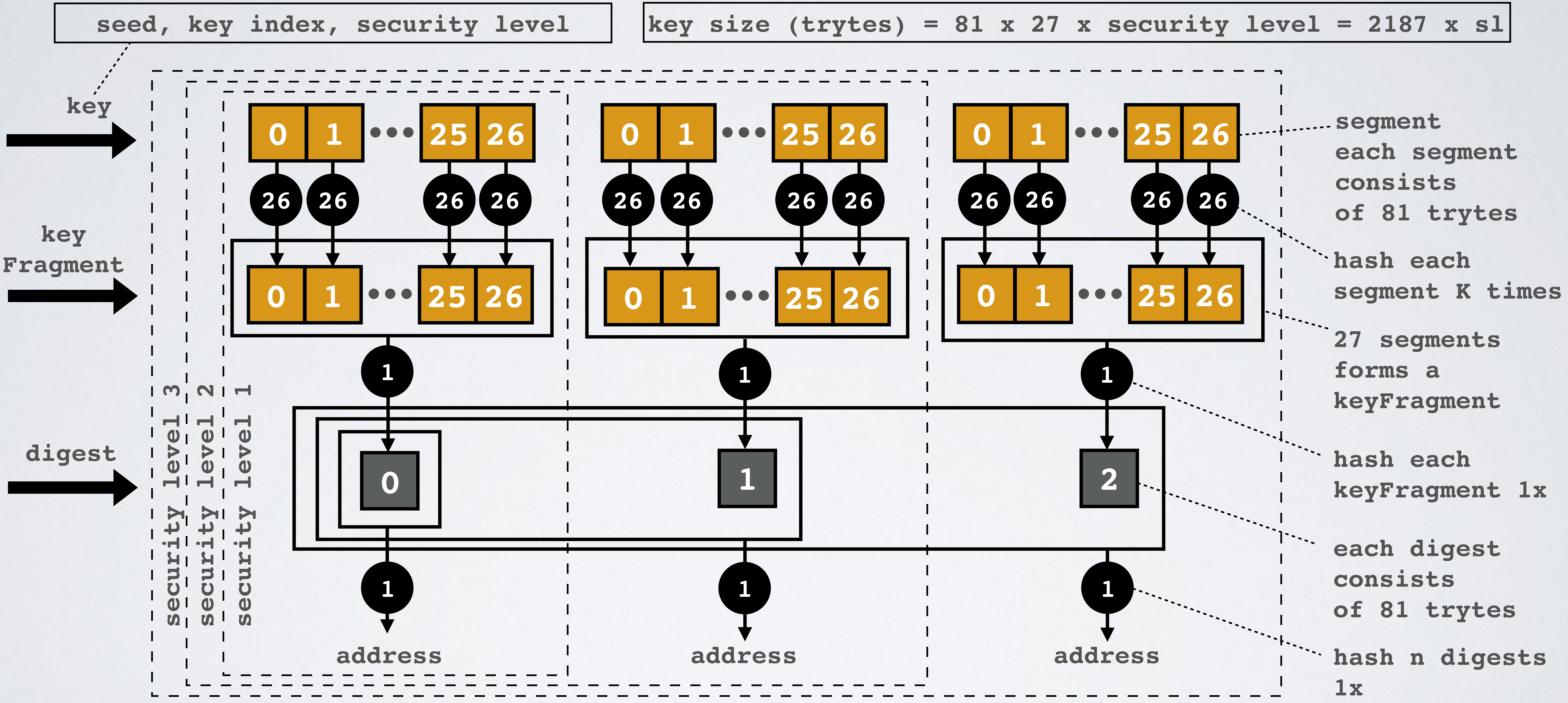
Kerl (Keccak-384)

absorb squeeze

address (public key)

address size (trits) = 243
address size (trytes) = 81

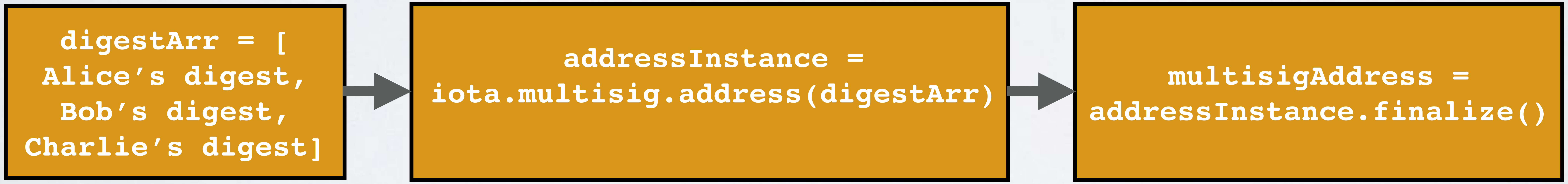
CALCULATE DIGEST


seed, key index, security level
key size (trytes) = 81 x 27 x security level = 2187 x sl
key
key Fragment
digest
security level 3
security level 2
security level 1
0
1
25
26
26
1
0
2
address
segment
each segment consists of 81 trytes
hash each segment K times
27 segments forms a keyFragment
hash each keyFragment 1x
each digest consists of 81 trytes
hash n digests 1x

MULTISIGNATURE PROCESS: MULTISIG ADDRESS

- Bob and Charlie can safely send their digest to Alice because Alice can not reconstruct their private keys using their digests.

- Alice, in her role as organiser, uses all three digests (3-of-3 scheme) to create a multisignature address.



- API to create the multisig address:
addressInstance = iota.multisig.address(digestArr)
multisigAddress = addressInstance.finalize()

MULTISIGNATURE PROCESS: MULTISIG ADDRESS

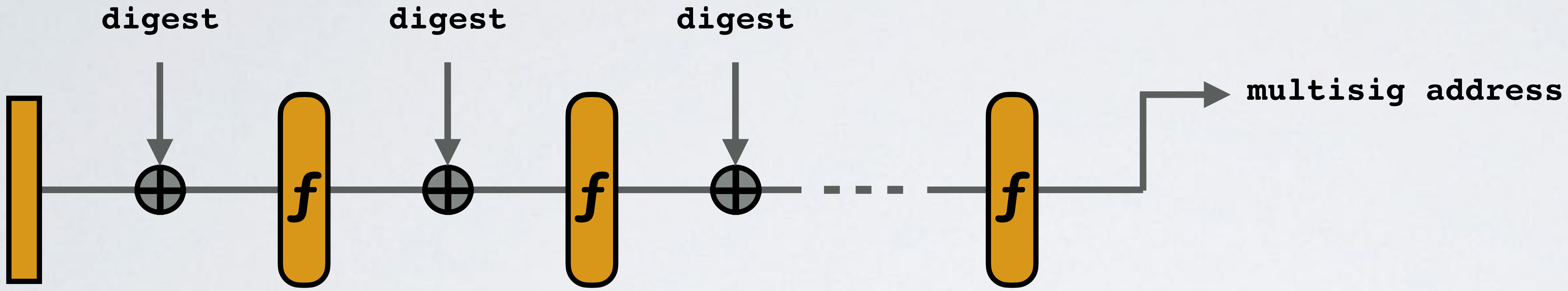
- The digest order is very important! See digestArr.

- Later, when the bundle needs to be signed the same order must be used.
  In this example the order is: Alice, Bob and Charlie.

- There is no difference between a multisignature address and an ordinary address.

- The size of a multisignature address and an ordinary address are both 81 trytes or 90 trytes with checksum.

CALCULATE MULTISIG ADDRESS



```
digest size (trits)    = 243 x security level
digest size (trytes)   = 81  x security level

multisig address size (trits)  = 243
multisig address size (trytes) = 81
```

MULTISIGNATURE PROCESS: BUNDLE

- Alice has created the multisig address and sends this address to Bob and Charlie.

- Bob and Charlie can now each deposit 1 IOTA to this multisig address.

- Alice verifies if the multisig address contains 2 IOTAs.

- After the verification Alice creates an initial bundle (initialBundle).

- In this particular example, the complete amount in the multisig address will be sent to the recipient address which means no remainder address is needed.

MULTISIGNATURE PROCESS: BUNDLE

- API to generate the initial bundle:
  iota.multisig.initiateTransfer(input, remainderAddress, transfers, function(err, initialBundle) {})

```
input = {
address: multisigAddress,
securitySum: sumSecurityLevels};

transfers = [{
address: recipientAddress,
value: amount,
message: message,
tag: tag}];
```

MULTISIGNATURE PROCESS: BUNDLE

- For a better understanding how the iota.multisig.initiateTransfer API works, see:
  IOTA Tutorial 15: BundleHash
  IOTA Tutorial 16: normalizedBundleHash

- See the generated initial bundle:
  https://www.mobilefish.com/download/iota/multisig_demo.txt

- Please note: There are 6 (sumSecurityLevels) transaction bundle entries where the address equals the multisig address. In these entries the signatureMessageFragment are empty (all 9's).

MULTISIGNATURE PROCESS: ADD SIGNATURES

- The initial bundle can now be signed.
  The signing order (Alice, Bob and last Charlie) is the same as the digest array order.

- First Alice adds her signature to the initial bundle.
  The modified bundle is now called firstSignedBundle.

- Alice sends the firstSignedBundle to Bob and he adds his signature to the bundle.
  The modified bundle is called secondSignedBundle.

- Bob sends the secondSignedBundle to Alice. Alice checks if the relevant signatureMessageFragment fields are modified.

- Alice sends the secondSignedBundle to Charlie and he adds his signature to the bundle. The modified bundle is called thirdSignedBundle.

MULTISIGNATURE PROCESS: ADD SIGNATURES

- Charlie sends the thirdSignedBundle to Alice. Alice checks if the relevant signatureMessageFragment fields are modified.

- See the modified firstSignedBundle, secondSignedBundle and thirdSignedBundle: https://www.mobilefish.com/download/iota/multisig_demo.txt

- Please note: The 6 (sumSecurityLevels) transaction bundle entries where the address equals the multisig address all have non-empty signatureMessageFragment values.

MULTISIGNATURE PROCESS: ADD SIGNATURES

- API to add signature to the bundle:
  privateKey = iota.multisig.getKey(seed, keyIndex, securityLevel);

  iota.multisig.addSignature(bundle, multisigAddress, privateKey, function(err, bundle) {…});

- For a better understanding how the iota.multisig.addSignature API works, see:
  IOTA Tutorial 17: Create and validate a signature

MULTISIGNATURE PROCESS: SEND BUNDLE TO TANGLE

- The thirdSignedBundle is the final bundle and Alice sends this bundle to the Tangle.

- API to send the bundle to the Tangle:
  iota.api.sendTrytes(trytes, depth, minWeightMagnitude, callback);

MULTISIGNATURE PROCESS: SEND BUNDLE TO TANGLE

```
const parsedFinalBundle = JSON.parse(finalBundle);

let trytes = [];

parsedFinalizedBundle.forEach(function(tx) {

   trytes.push(iota.utils.transactionTrytes(tx))

});

iota.api.sendTrytes(trytes.reverse(), depth, minWeightMagnitude, function(err, attached) {

   attached.forEach(function(tx) {

      console.log(JSON.stringify(tx, null, "\t"));

   });

});
```

M-OF-N SCHEME

- What if Alice decided to use a 2-of-3 signature scheme?
  You always need the signatures of ALL participants in the scheme.

- This means, according to the documentation
  https://github.com/iotaledger/wiki/blob/master/multisigs.md
  an M-of-N scheme can be setup by sharing private keys.

- Example 1: A 2-of-3 scheme means each participant in the scheme should share its private key with (N - M = 3 - 2 =) 1 other participant.

- Example 2: A 2-of-5 scheme means each participant in the scheme should share its private key with (N - M = 5 - 2 =) 3 other participants.

M-OF-N SCHEME

- Example 3: A 2-of-4 scheme means each participant in the scheme should share its private key with (N - M = 4 - 2 =) 2 other participants.

- So, if Alice wants to use of 2-of-3 scheme:
  Alice gives her private key to **B**ob (A-B).
  Bob gives his private key to **C**harlie (B-C).
  Charlie gives his private key to **A**lice (C-A).

- This means in a 2-of-3 scheme, 2 participants will have all the necessary signatures in order for a transfer to be successful.

M-OF-N SCHEME

- M-of-N private key sharing examples:

1-of-3	2-of-3	2-of-4	3-of-4	2-of-5
A - BC	A - B	A - BC	A - B	A - BCD
B - CA	B - C	B - CD	B - C	B - CDE
C - AB	C - A	C - DA	C - D	C - DEA
		D - AB	D - A	D - EAB
				E - ABC

A(lice)
B(ob)
C(harlie)
D(ave)
E(ve)

M-OF-N SCHEME

- When setting up an M-of-N scheme keep an eye on the following:
  - The whole process must be transparent to ALL its participants, meaning each participant must receive and understand the complete M-of-N private key sharing scheme.
  - Sending the private key to participants must happen securely.
  - Users must share their private key with only the assigned participant(s) and no one else.