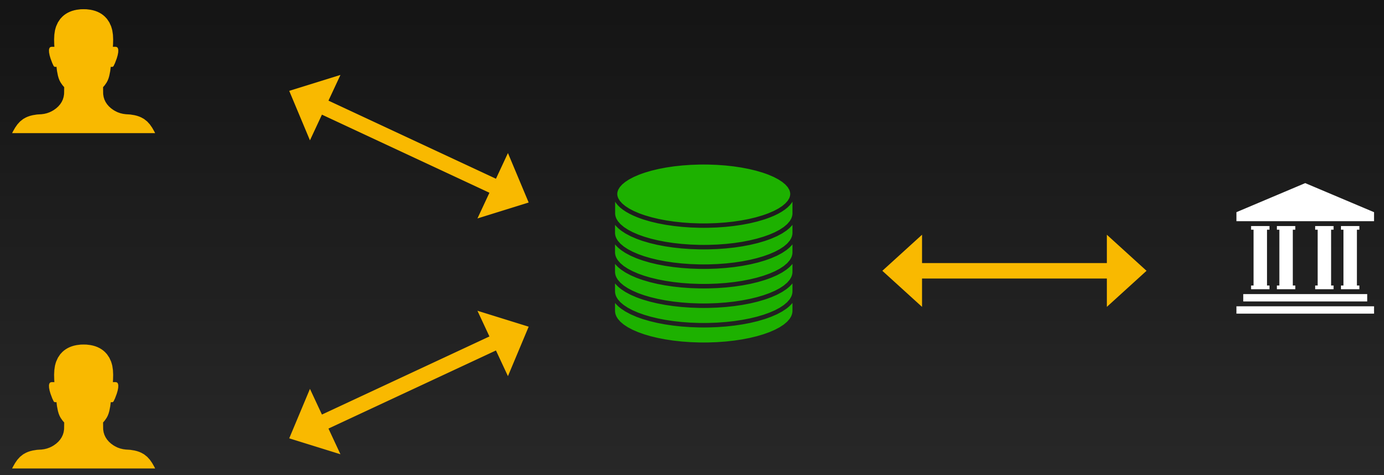
# FastPay

High-Performance Byzantine Fault Tolerant Settlement

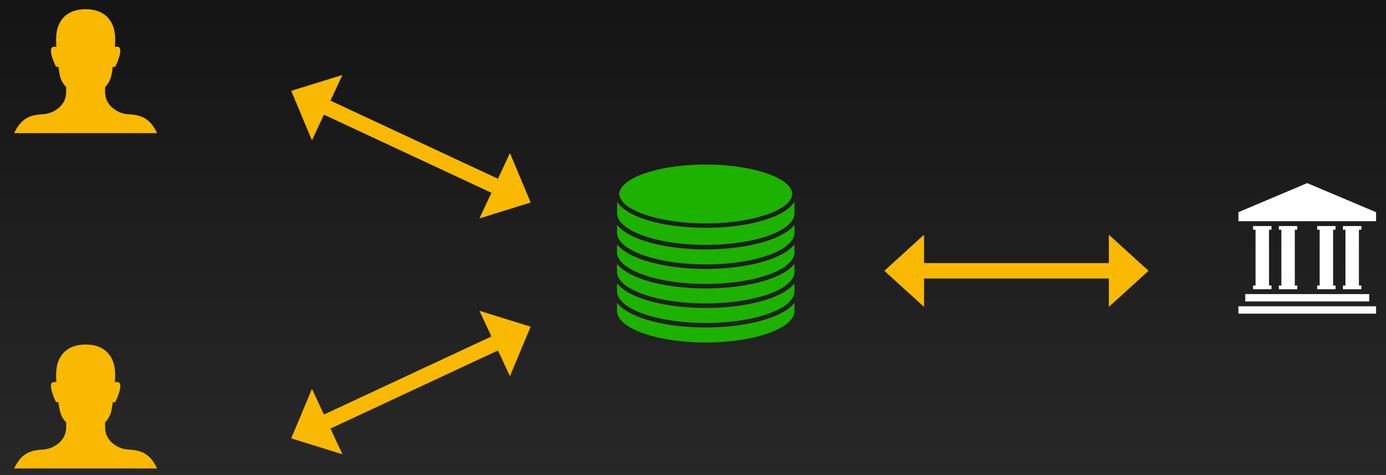
# RTGS

## A simplified view



# RTGS

## A simplified view

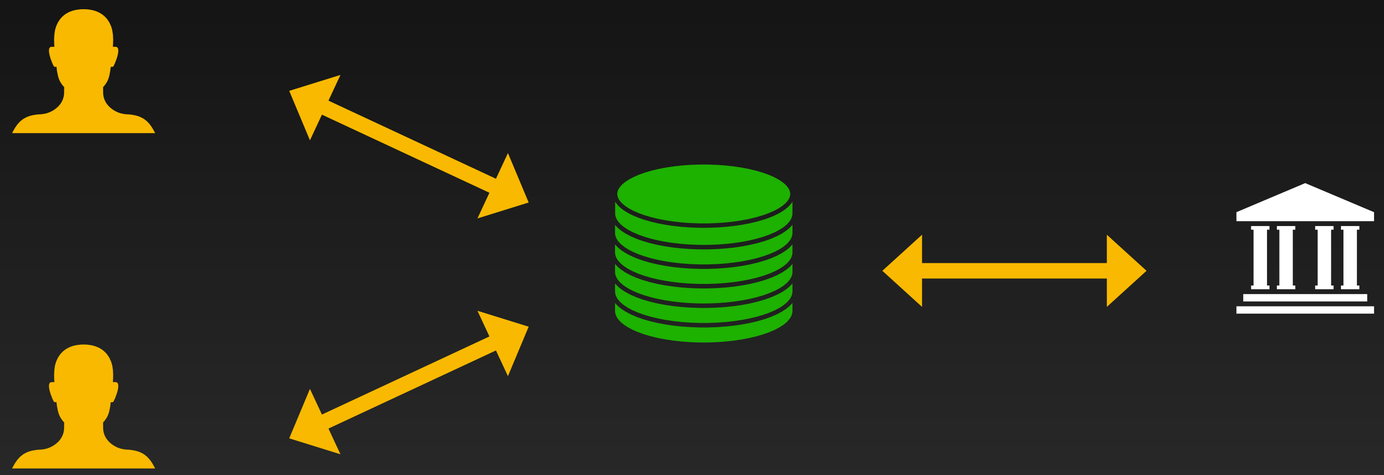


**Very centralized**

**Low capacity (expensive)**

# RTGS

## A simplified view

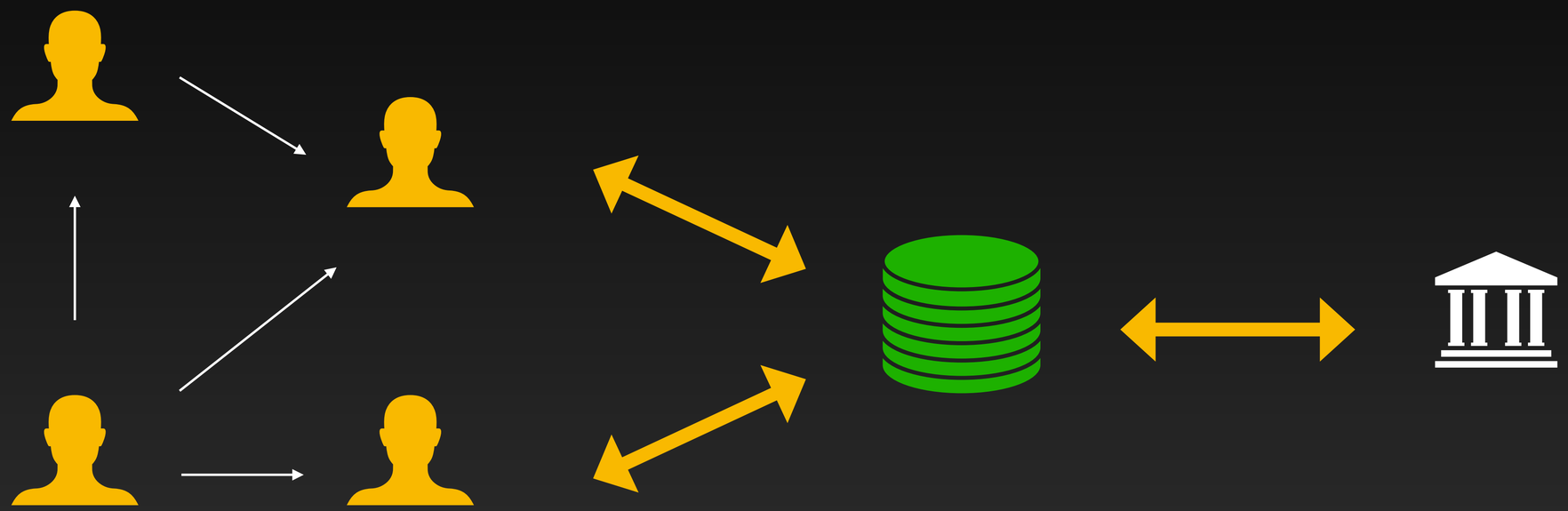


**TPS:** 500 tx/s

**Latency:** minutes

# RTGS

## A simplified view



**TPS:** 80,000 tx/s

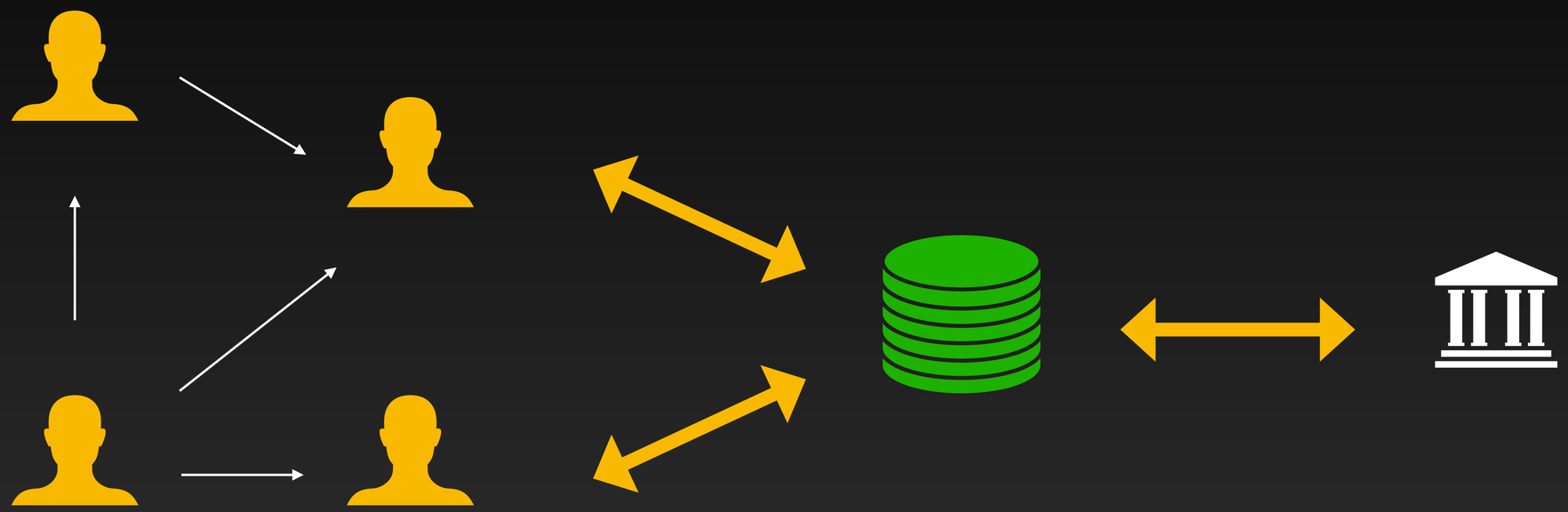
**Latency:** seconds

**TPS:** 500 tx/s

**Latency:** minutes

# RTGS

## A simplified view

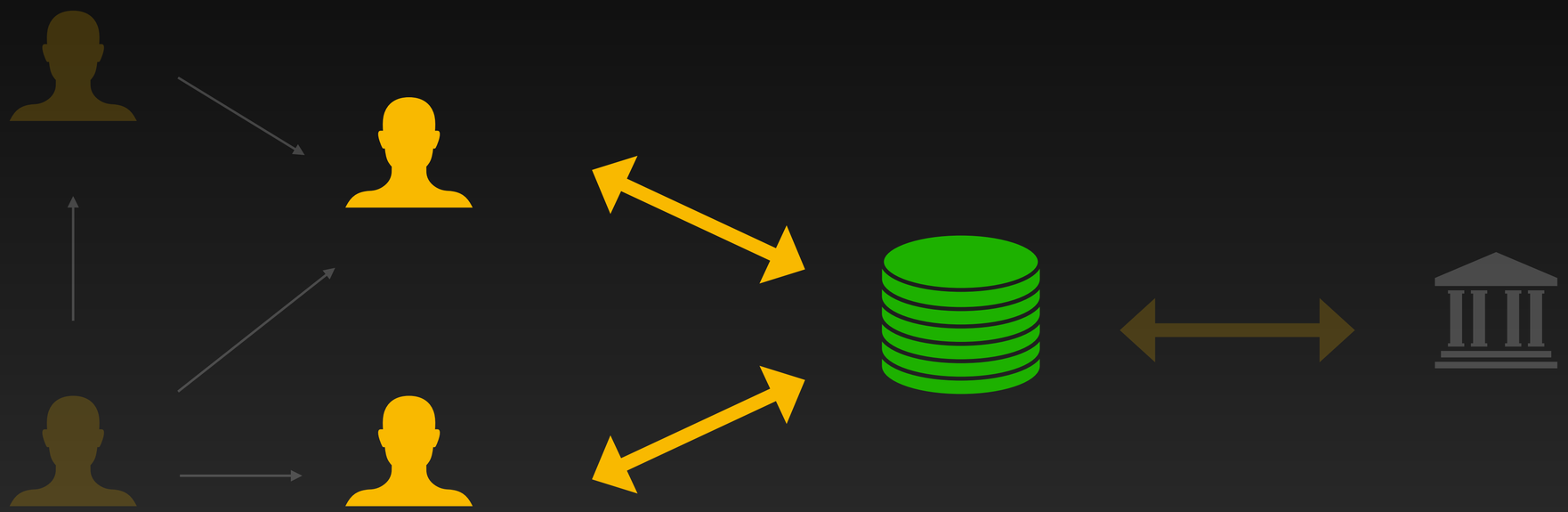


**Promises of payment**

**Settlement**

# RTGS

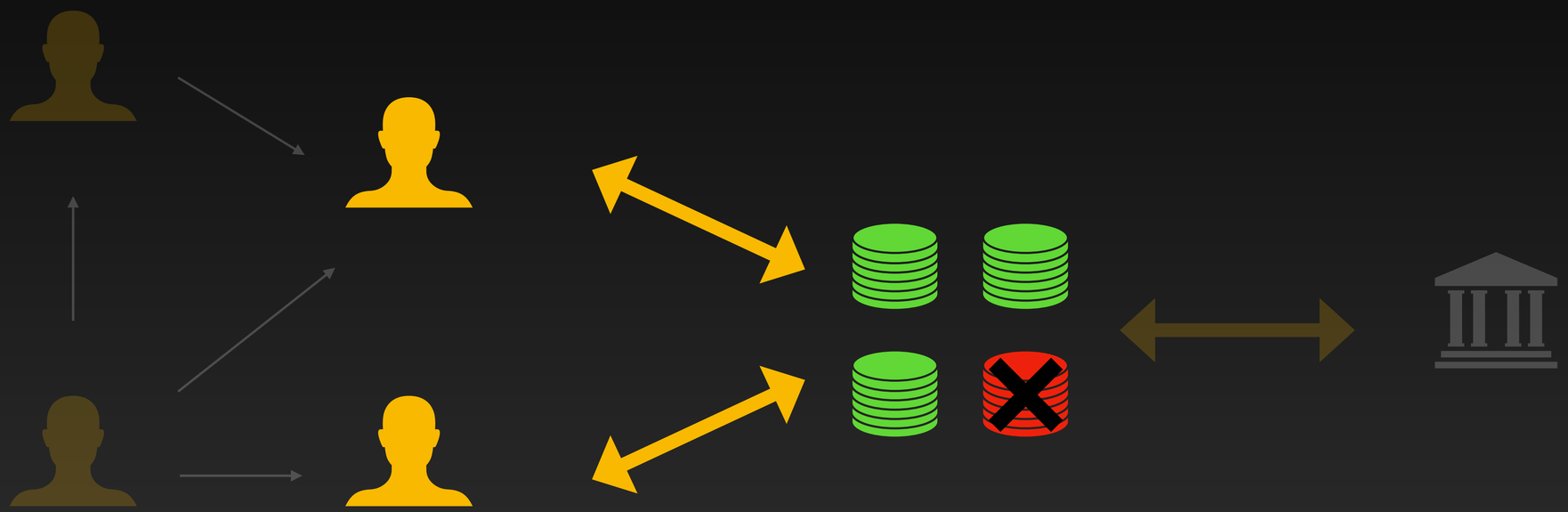
## A simplified view



**Fast settlement**

# RTGS

## A simplified view



**BFT resilience**

**High capacity (cheap)**

# Byzantine Fault Tolerance

> 2/3

# In summary

## What we want

- Low latency
- BFT reliance
- Fast finality
- Hight capacity

## Current industry

- Low latency (not settled)
- Centralized
- Slow finality
- Hight capacity (not settled)

# Make it practical for retail payment at physical points of sale

This requires extremely low latency

# FastPay

## Acknowledgments

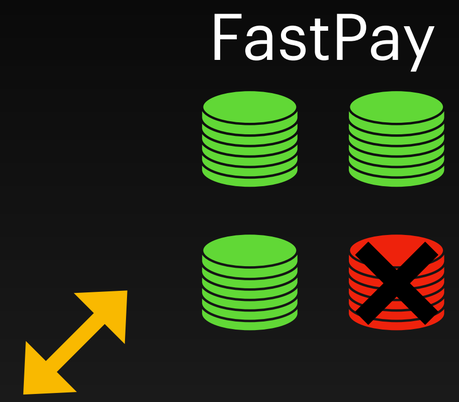
Mathieu Baudet

George Danezis

Alberto Sonnino

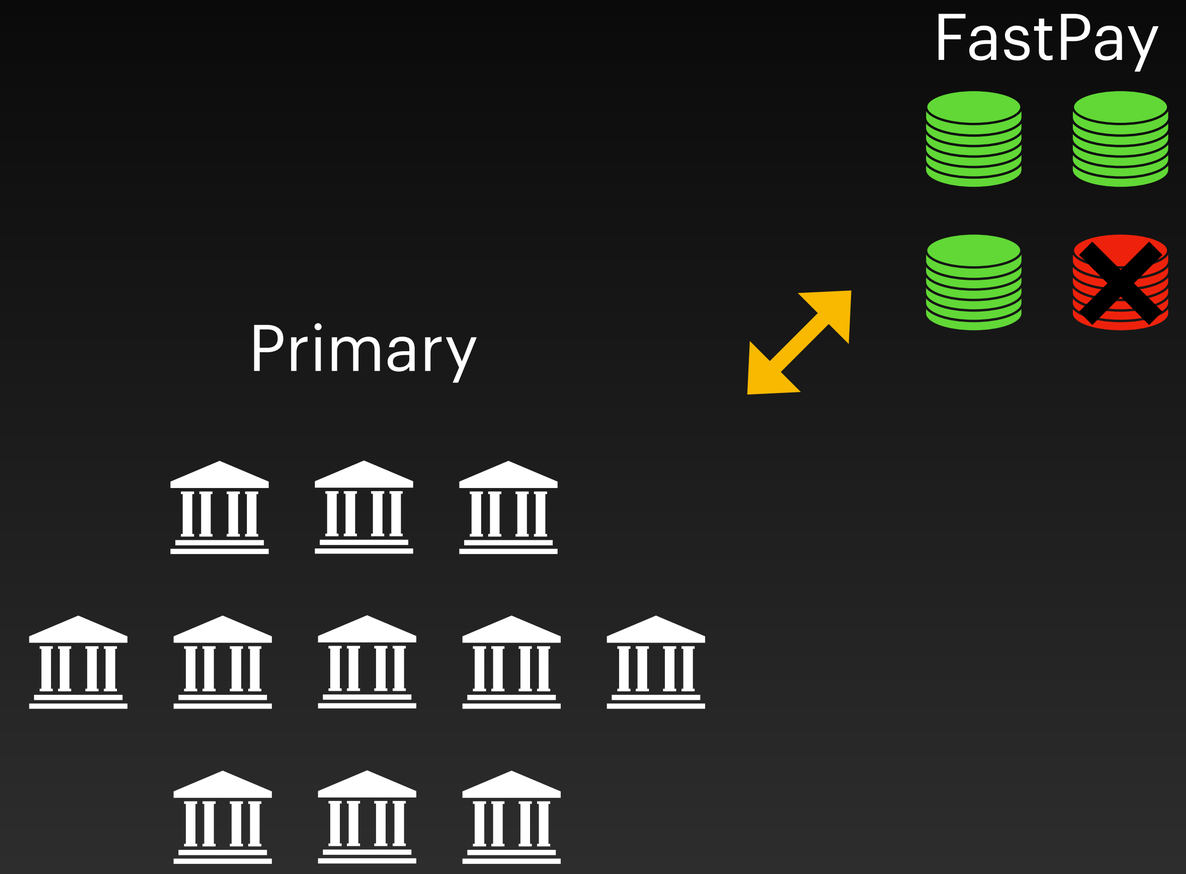
Facebook Novi

# Overview

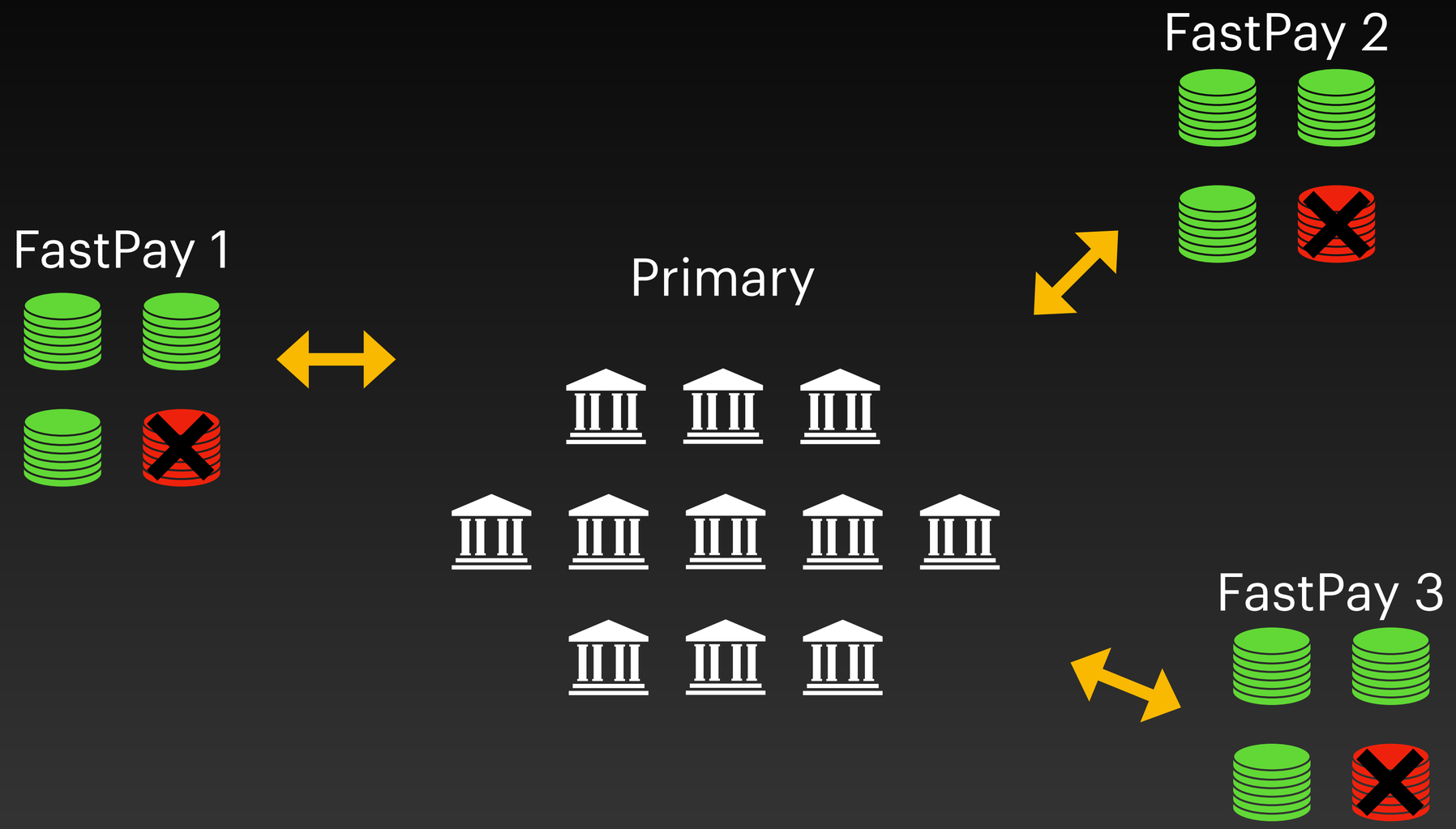



Primary

# Overview

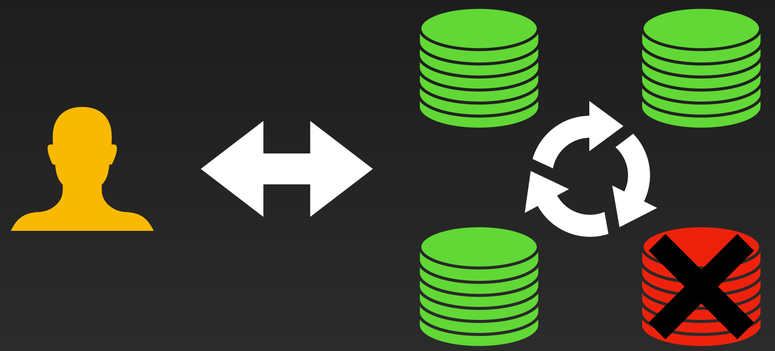


# Overview



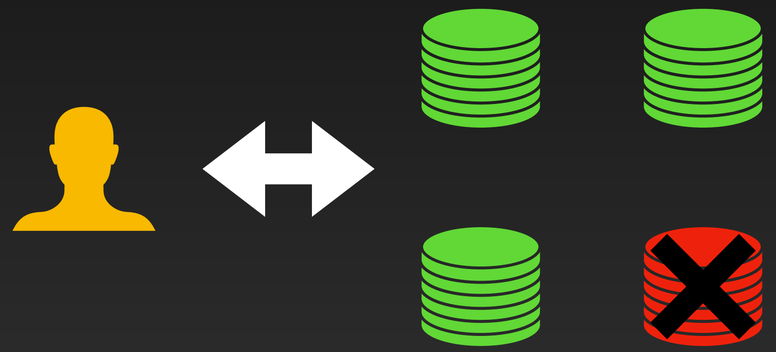
# Difference with blockchains

## Blockchains



Byzantine Consensus

## FastPay



Byzantine Consistent Broadcast

# FastPay

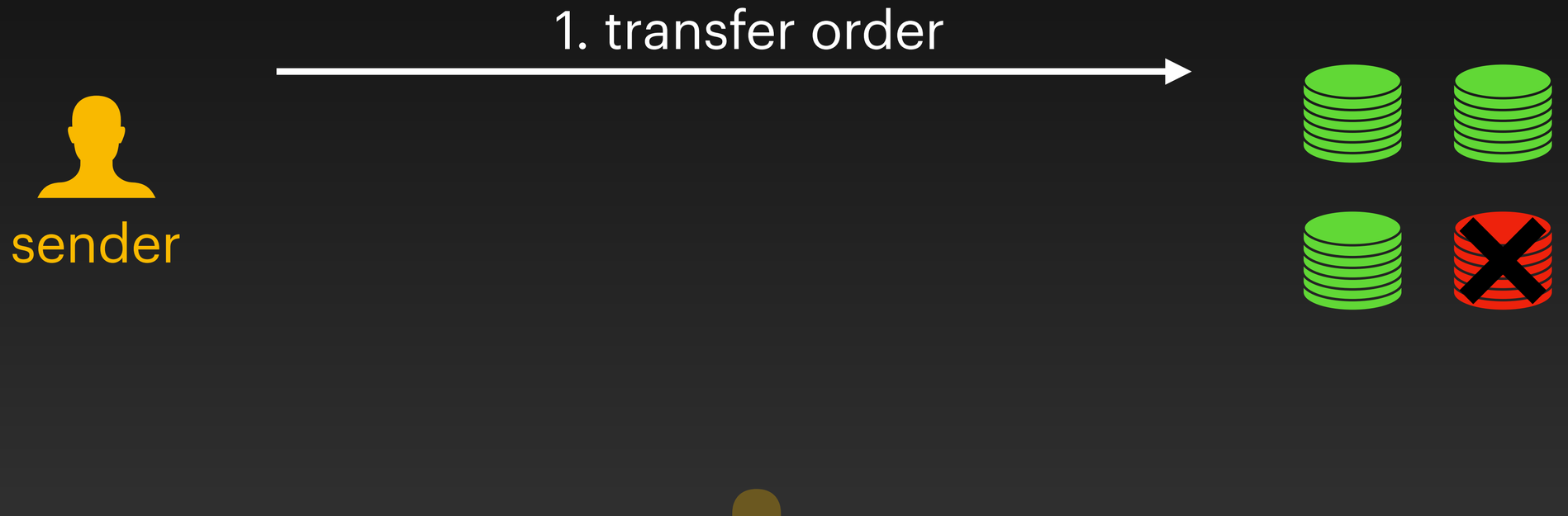
## How does it work?

sender

recipient

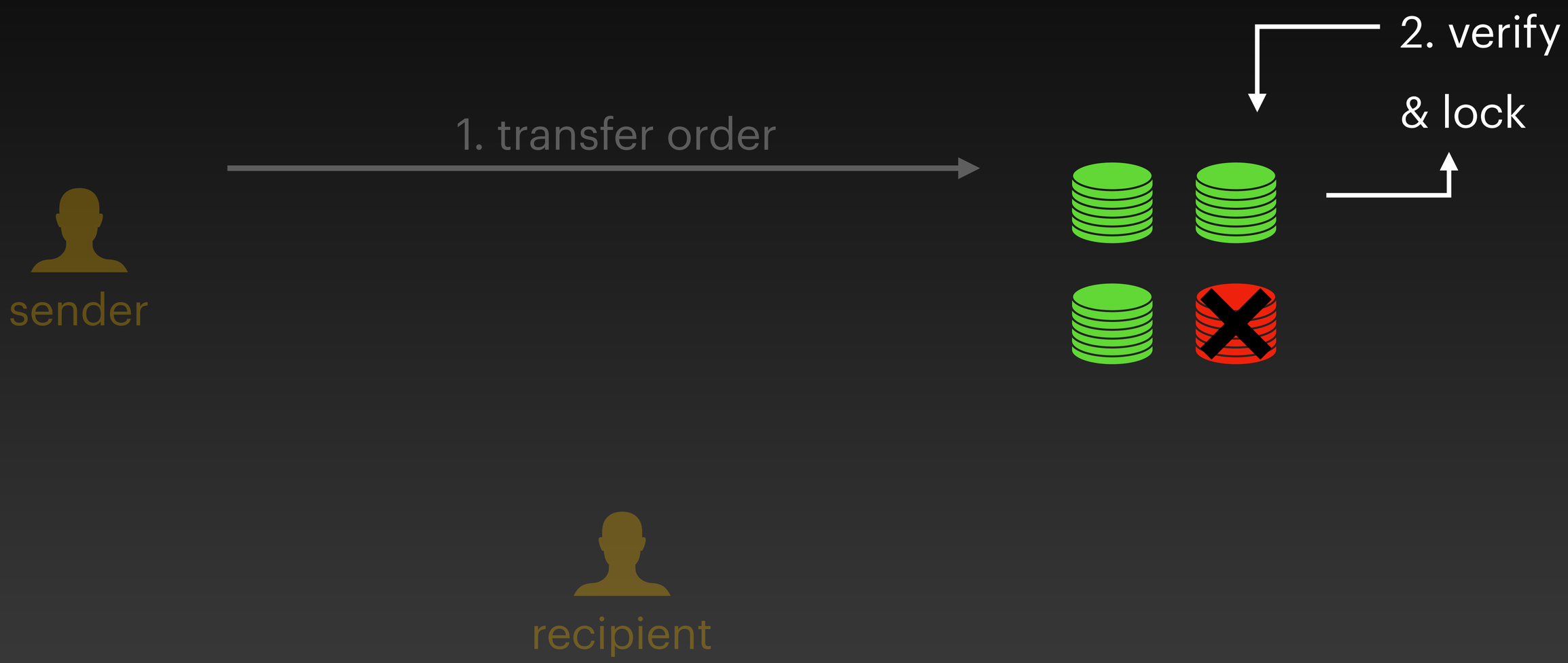
# FastPay

## How does it work?



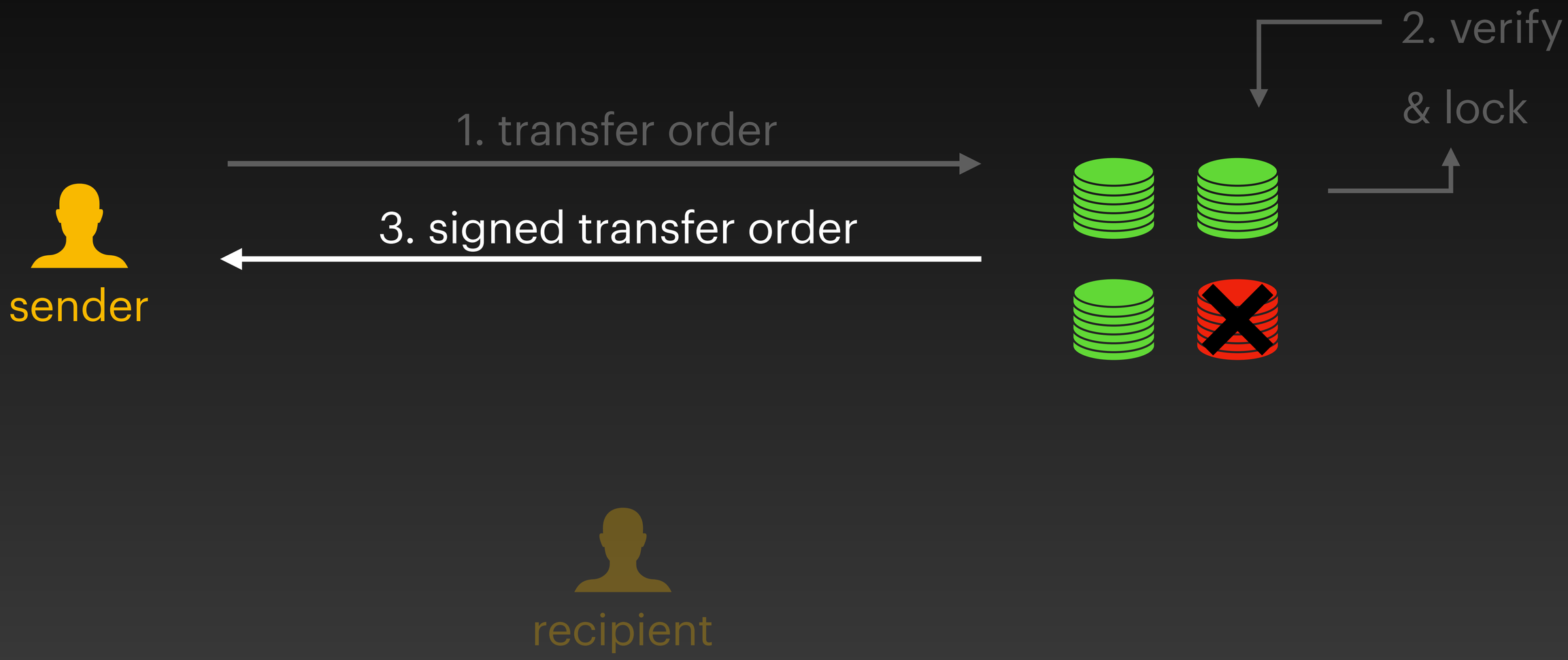
# FastPay

## How does it work?



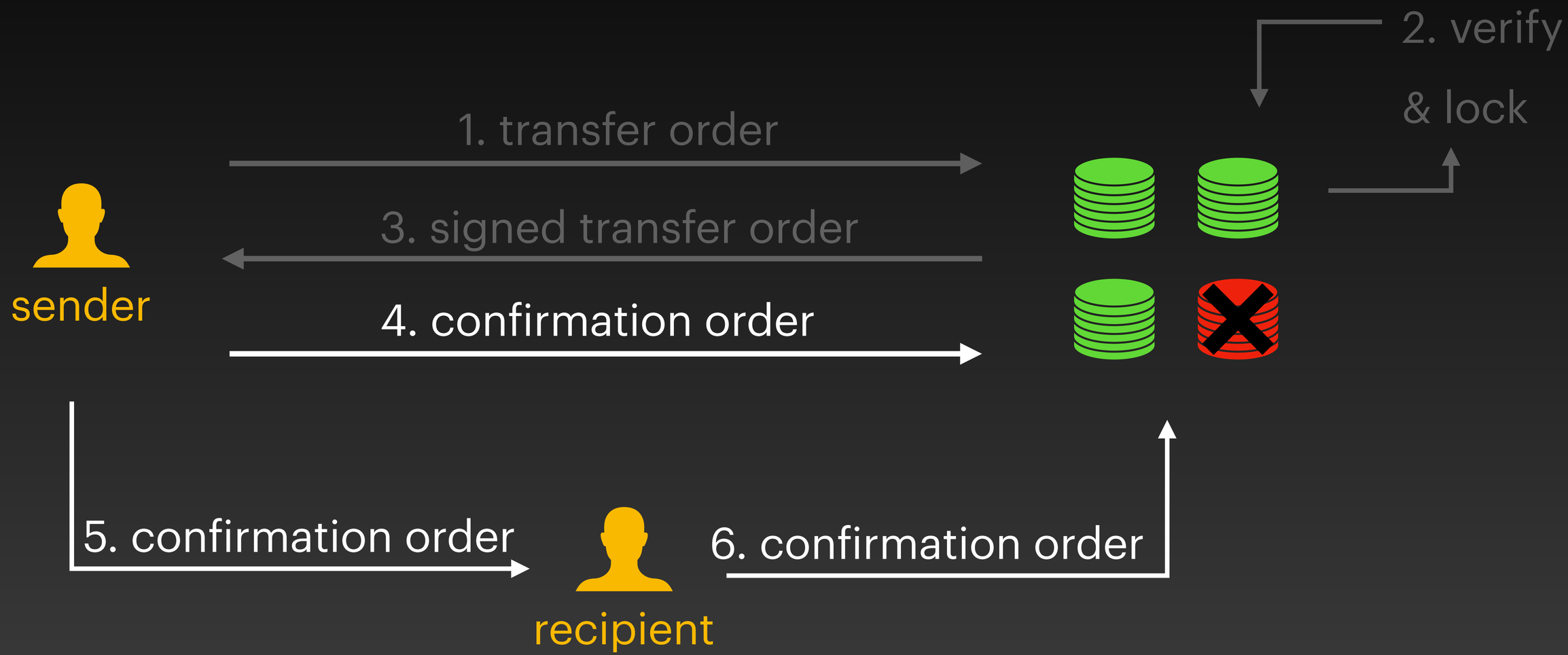
# FastPay

## How does it work?



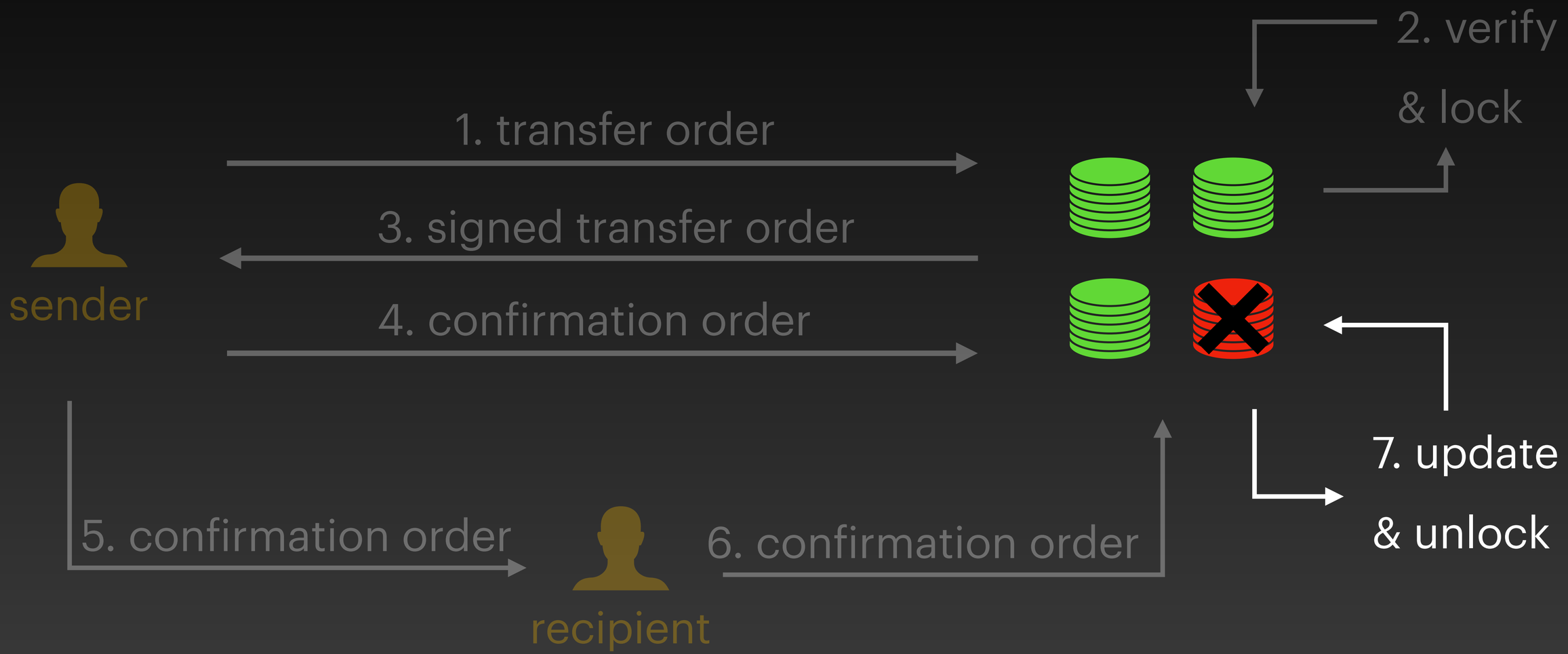
# FastPay

## How does it work?



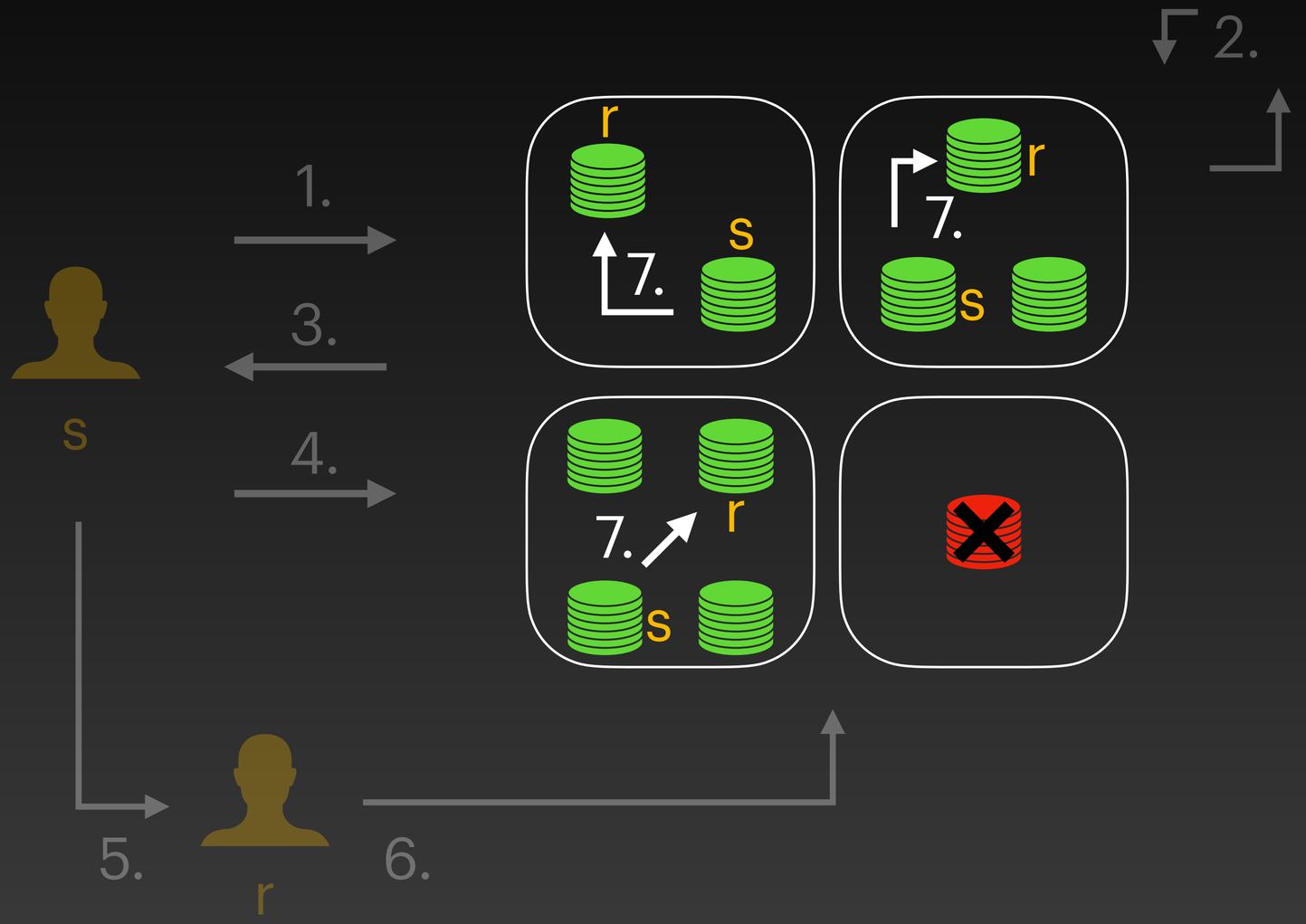
# FastPay

## How does it work?



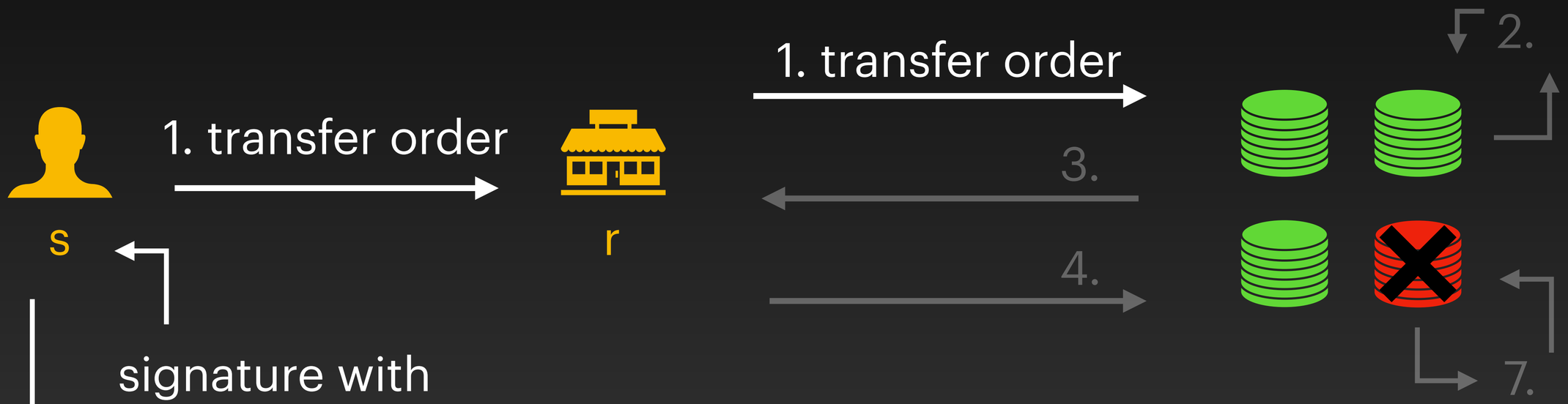
# FastPay

## Increasing capacity

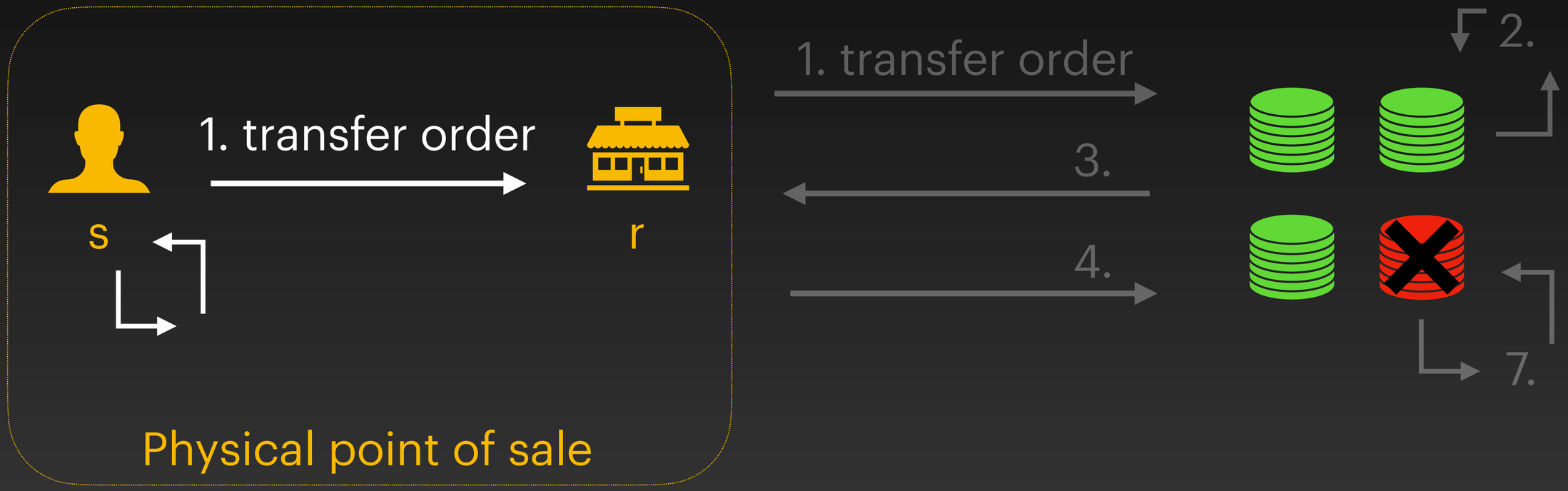


# FastPay

## Using proxies

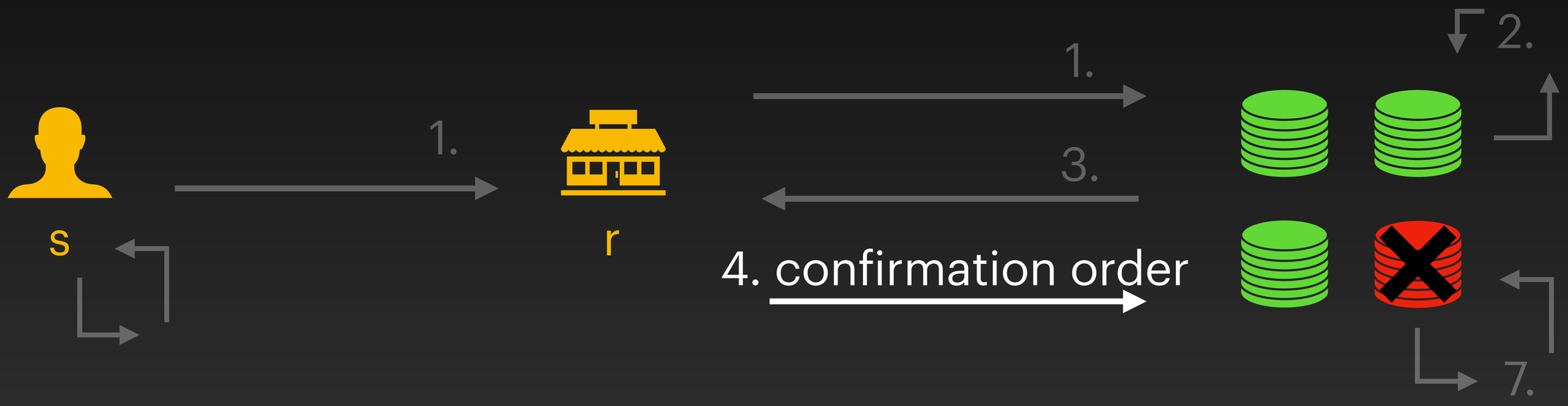


# FastPay
## Using proxies



# FastPay

## Using proxies




Anyone can aggregate and broadcast confirmation orders by reading the authorities' state

# Byzantine Consistent Broadcast

Validity

No duplication

Integrity

Consistency

# FastPay
## Authorities' state

### Authorities

- Authority name and keys
- Committee information
- Accounts information
- Last primary tx index

### Each account

- Verification key
- Balance
- Sequence number
- Last transfer order
- List of certificates and synchronization orders

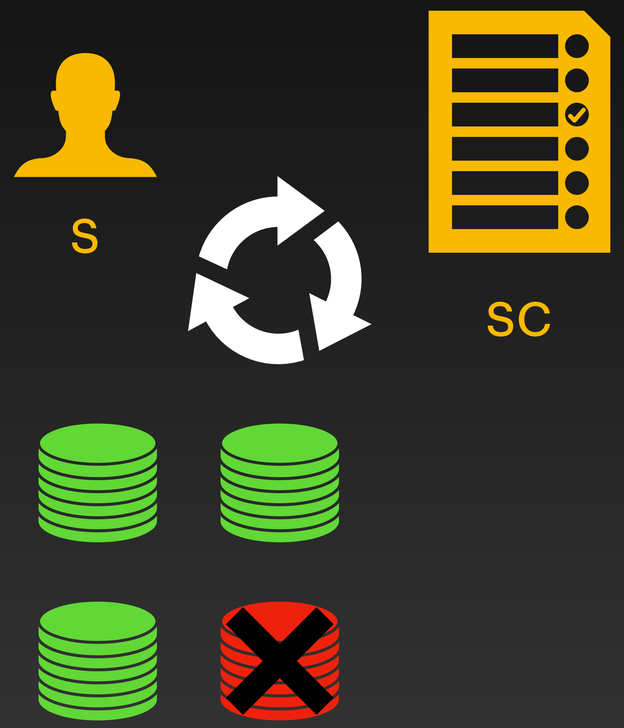
# FastPay

## Clients' state

- Their account's address
- Their secret key
- Committee information
- Last sequence number
- Last signed transfer order

# FastPay

## Interface it with a primary infrastructure

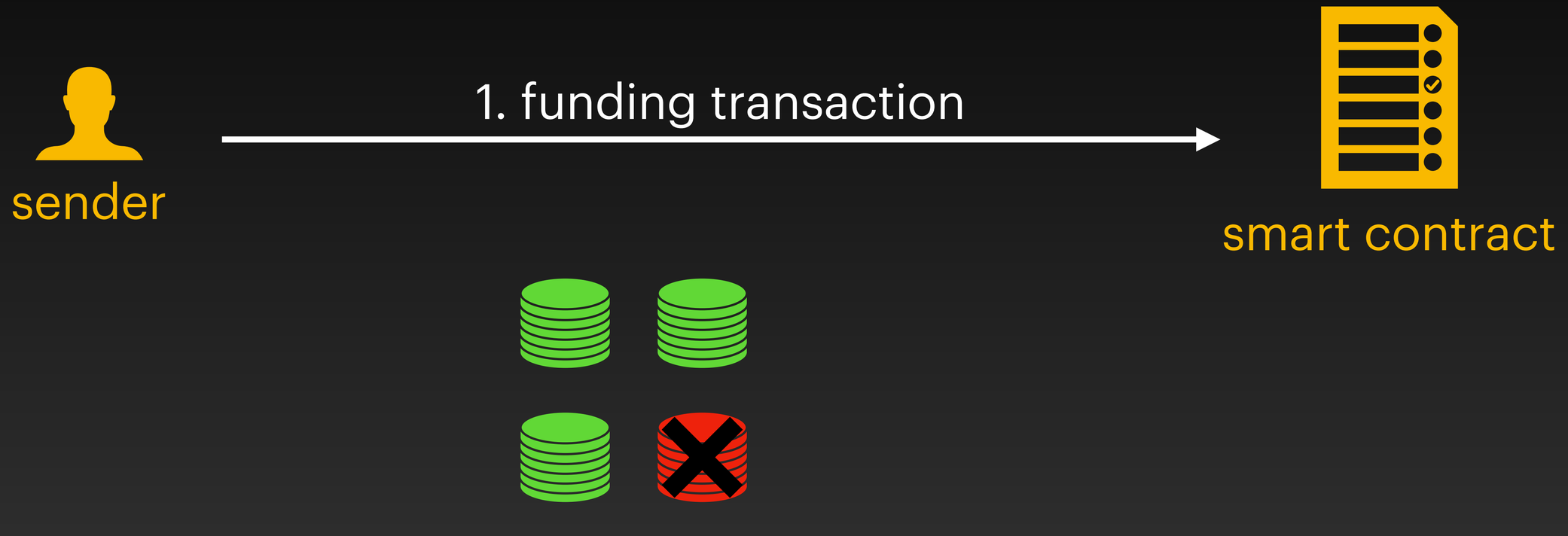



## Smart Contract's state

- The committee information
- Total funds in the contract
- Last primary tx index
- "Redeem log"

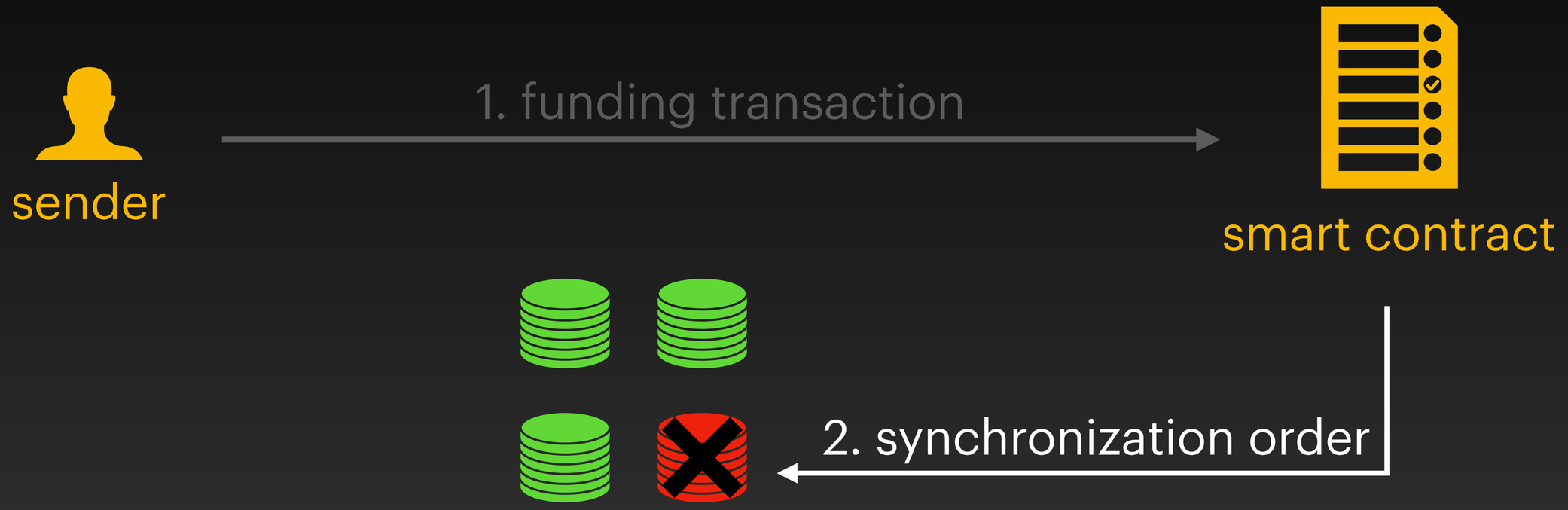
# FastPay

## From primary infrastructure to FastPay



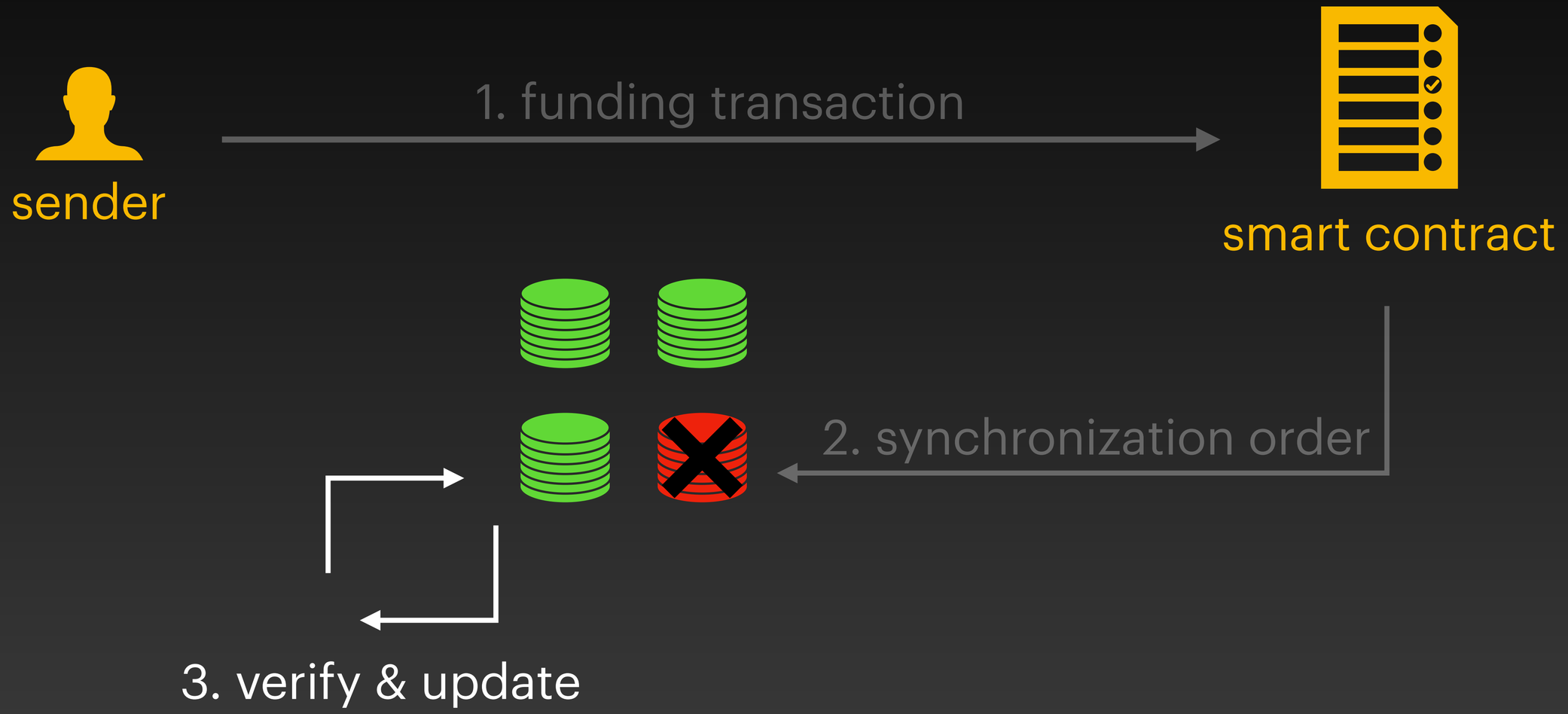
# FastPay

## From primary infrastructure to FastPay



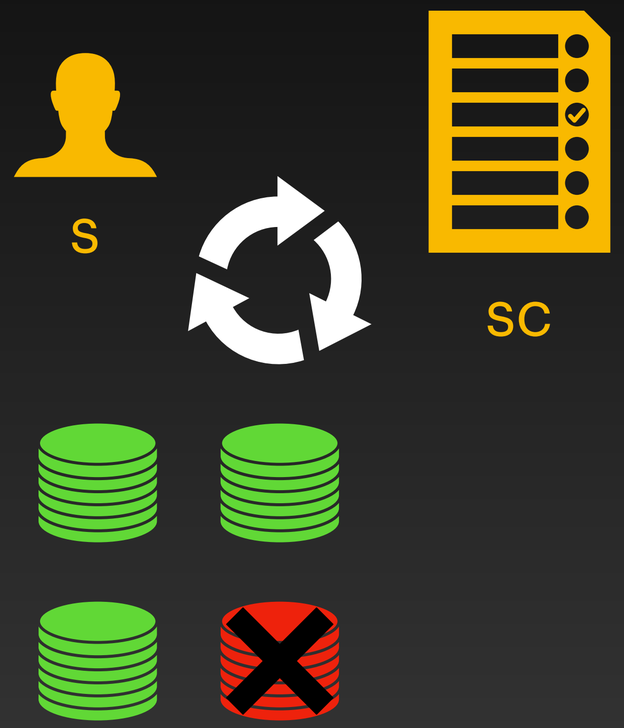
# FastPay

## From primary infrastructure to FastPay



# FastPay

## Interface it with a primary infrastructure

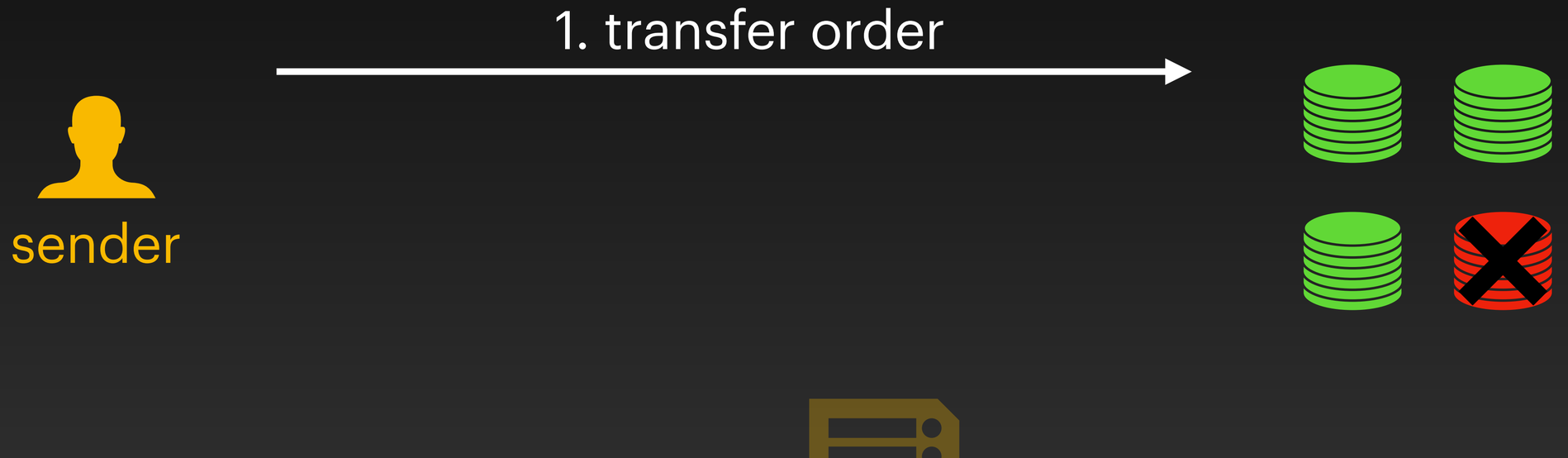



## Smart Contract's state

- The committee information
- Total funds in the contract
- Last primary tx index
- "Redeem log"

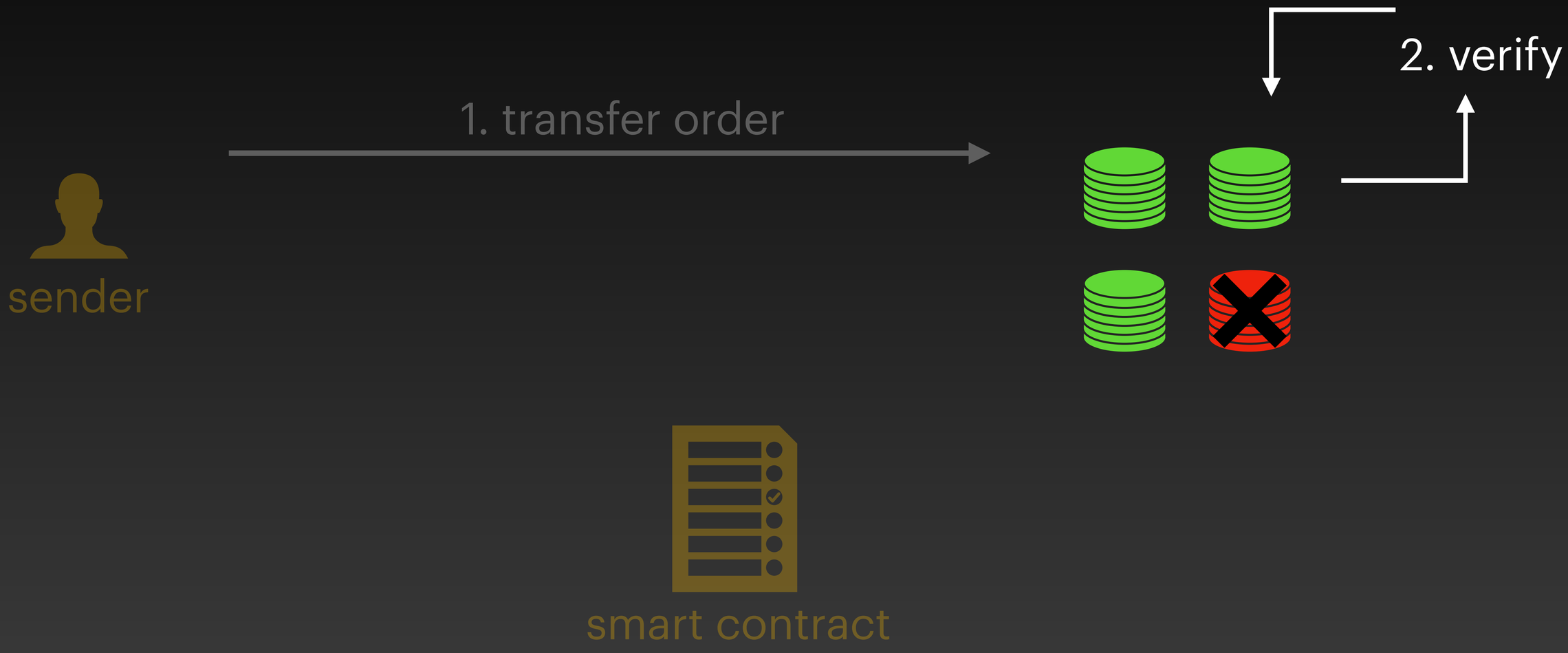
# FastPay

## From the primary infrastructure to FastPay



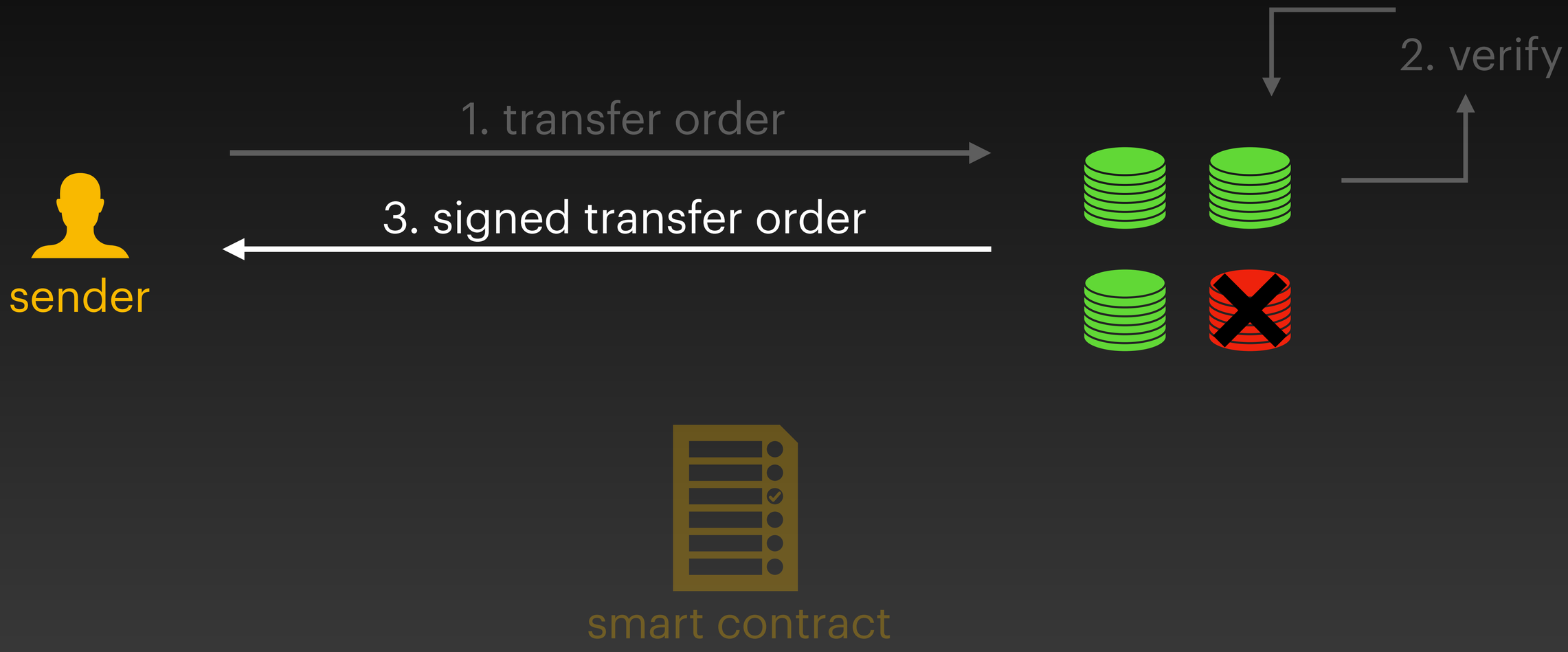
# FastPay

## From the primary infrastructure to FastPay



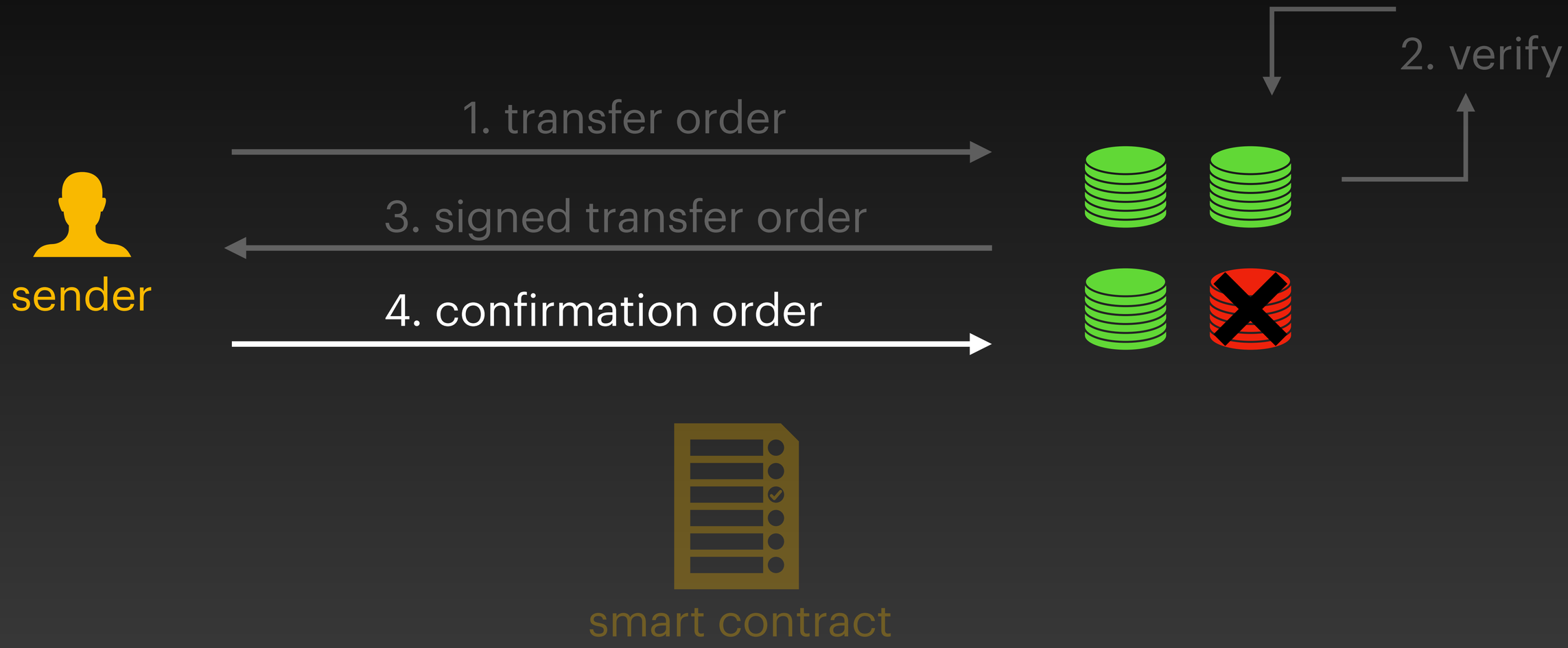
# FastPay

## From the primary infrastructure to FastPay



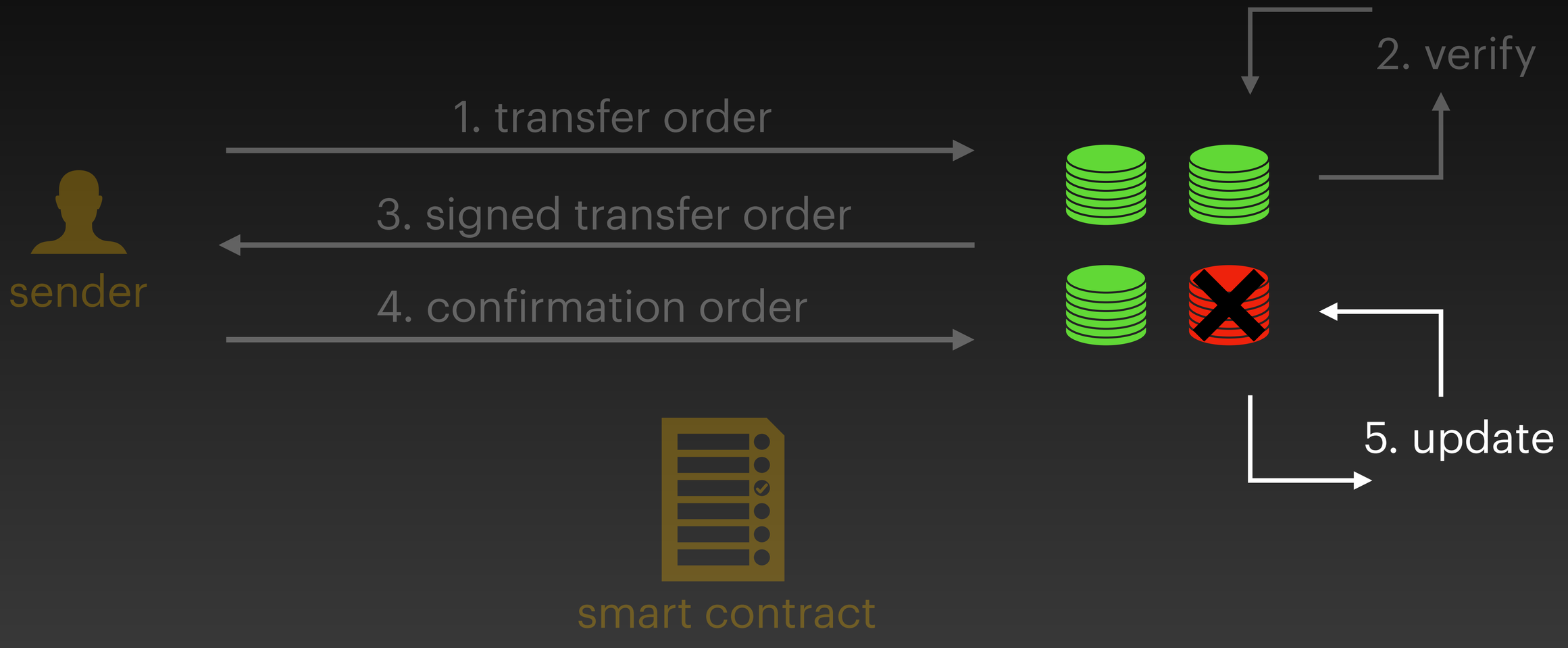
# FastPay

## From the primary infrastructure to FastPay



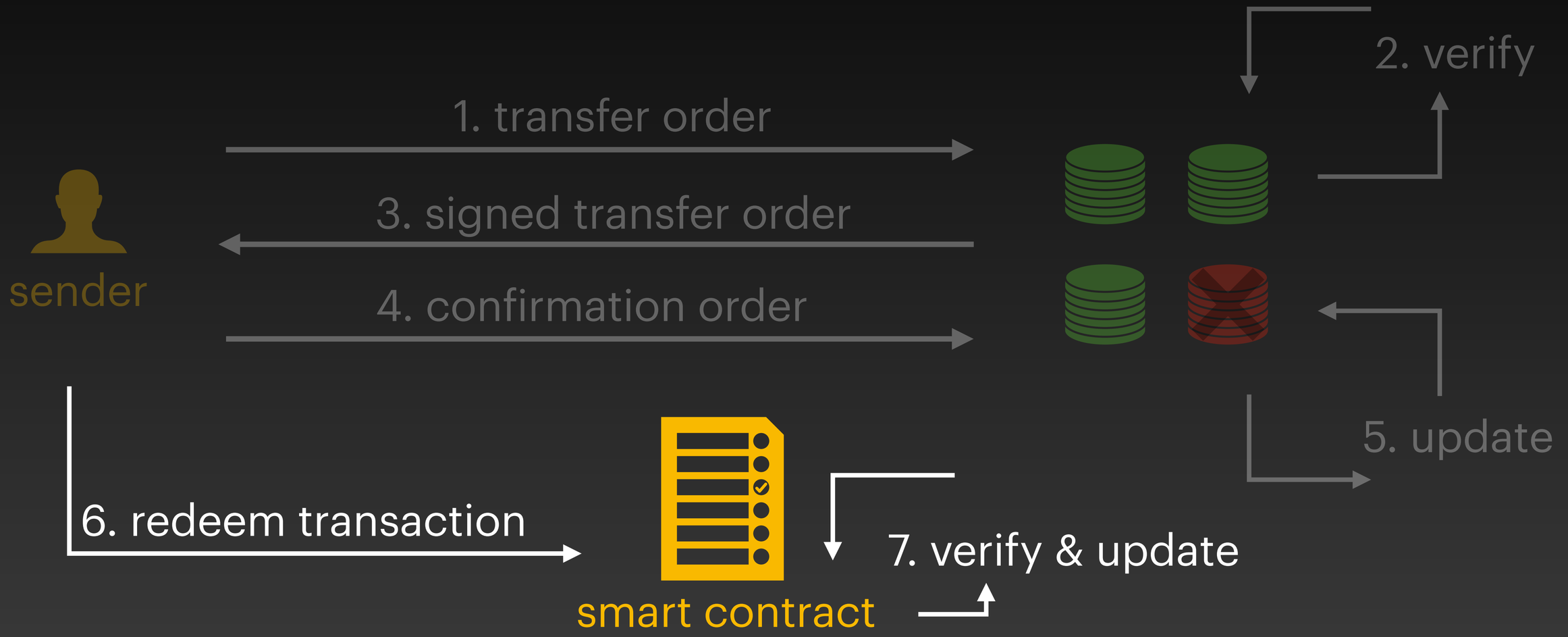
# FastPay

## From the primary infrastructure to FastPay



# FastPay

## From the primary infrastructure to FastPay



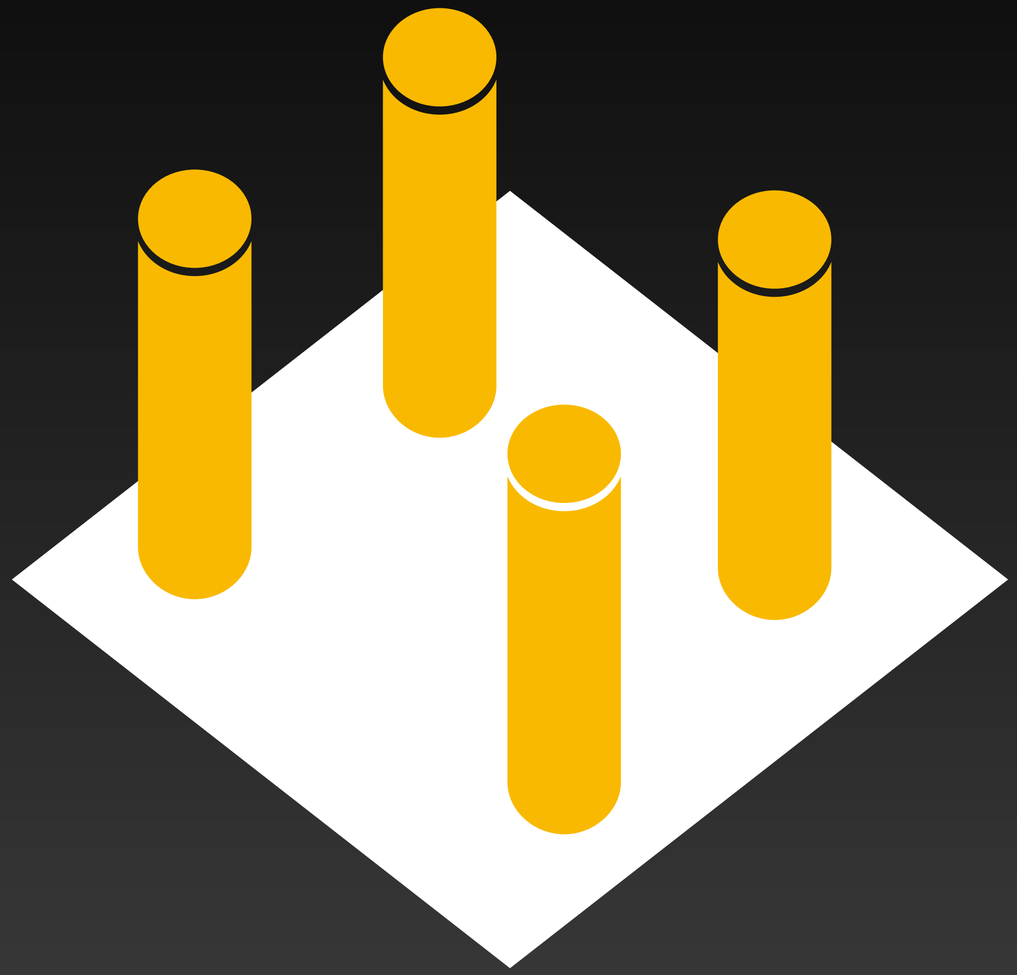
# FastPay
## Implementation

- Written in Rust
- Networking: Tokio & UDP
- Cryptography: ed25519-dalek

**https://github.com/novifinancial/fastpay**

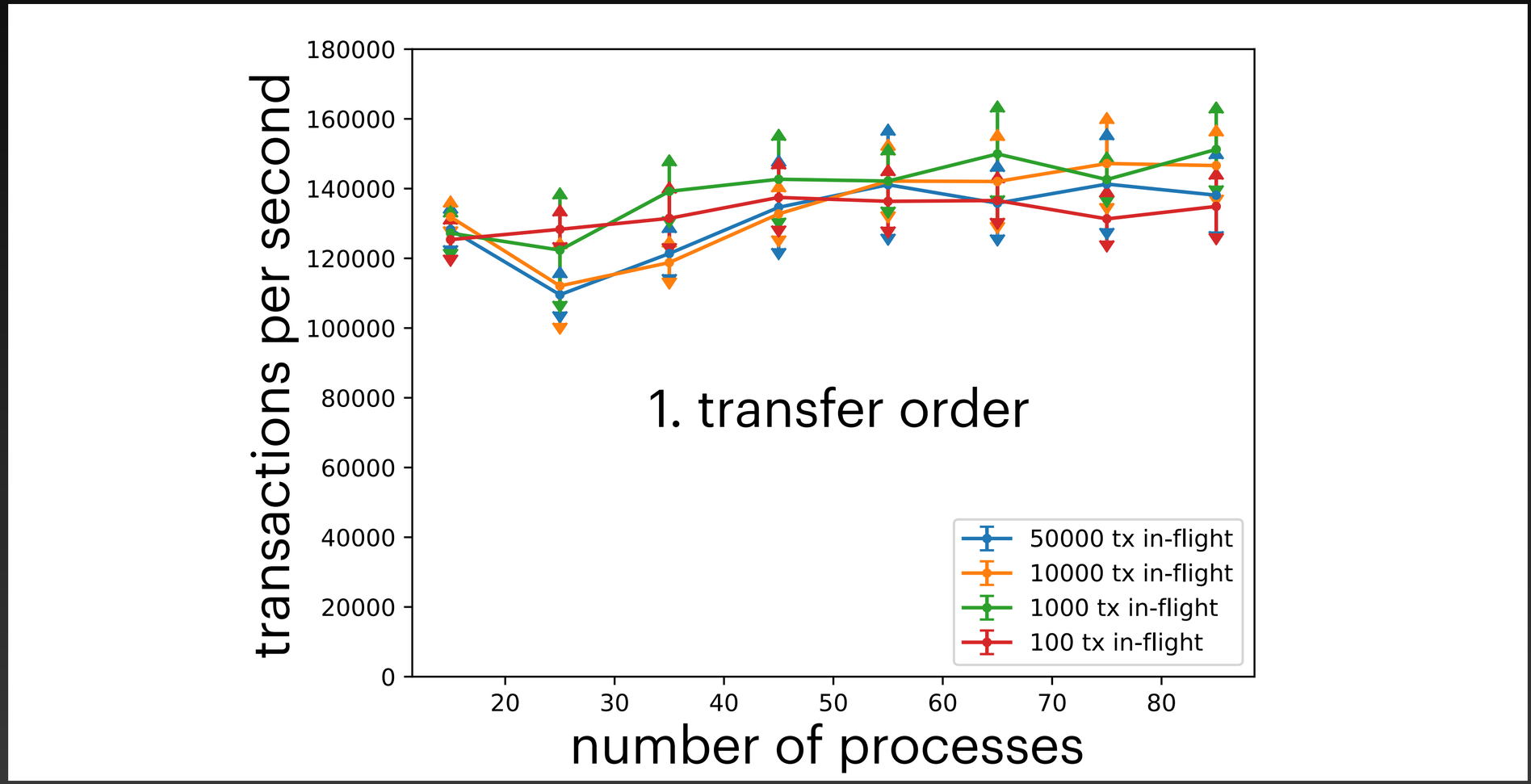
# FastPay

## Throughput Evaluation



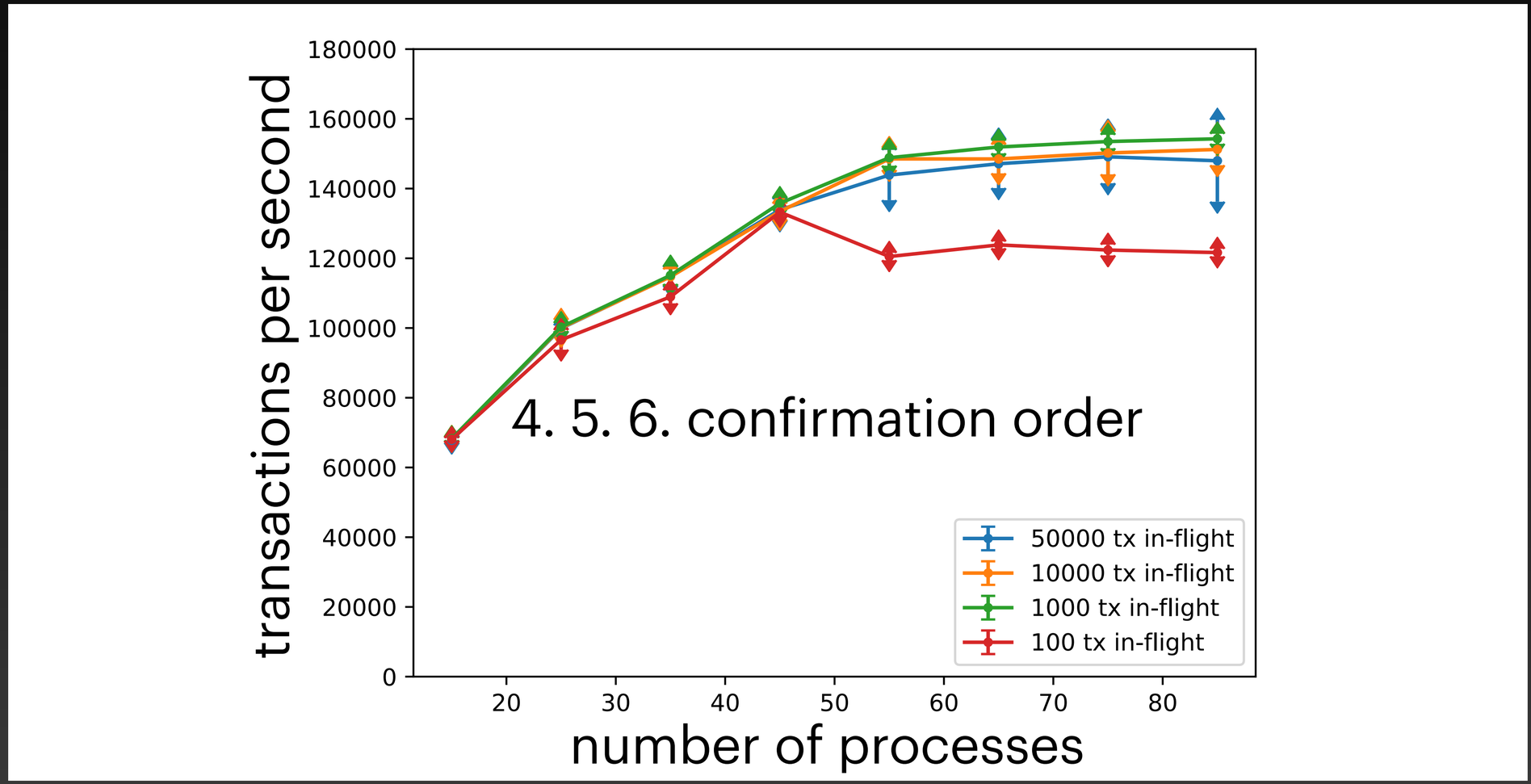
# FastPay

## High concurrency

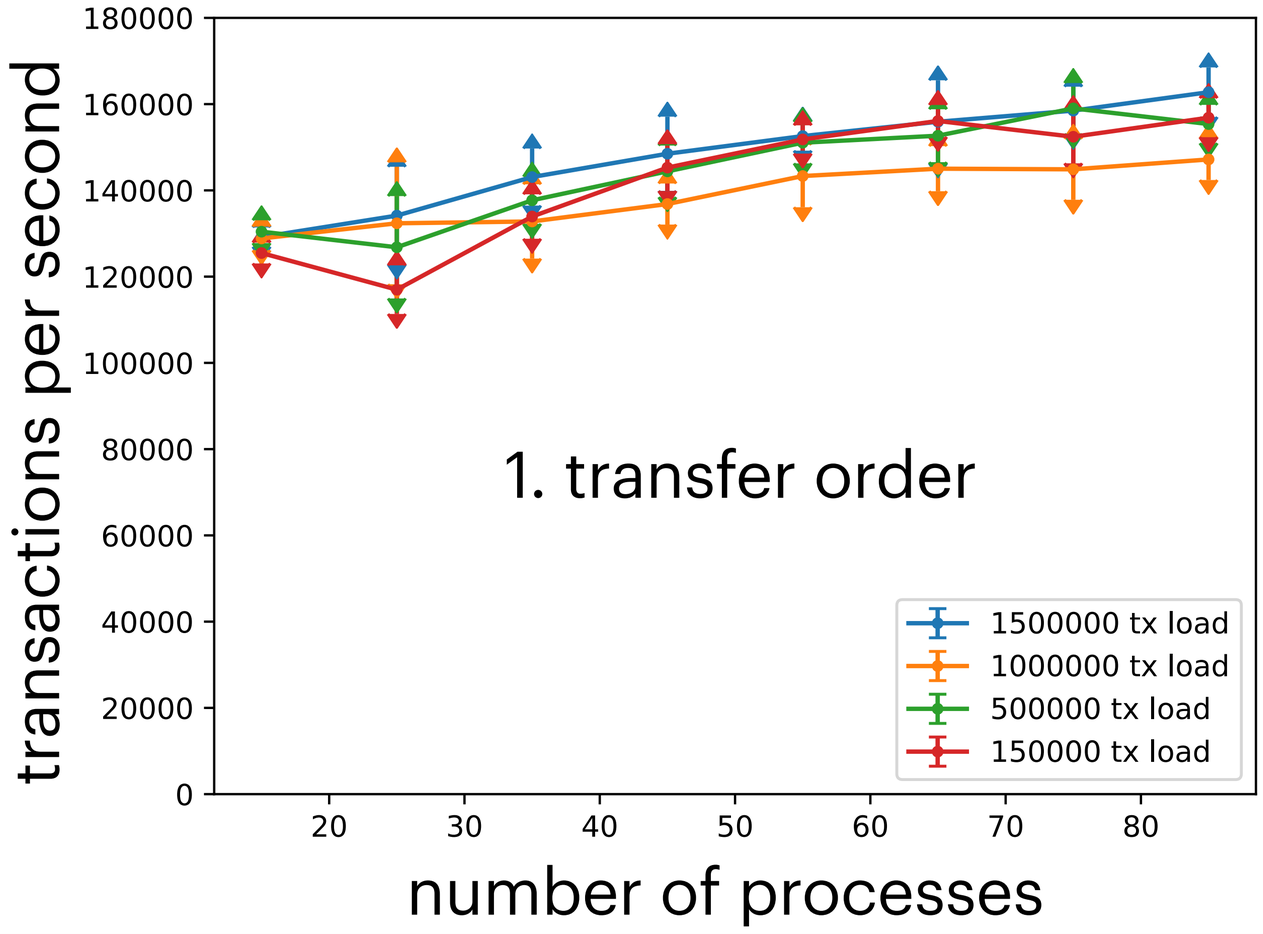


# FastPay

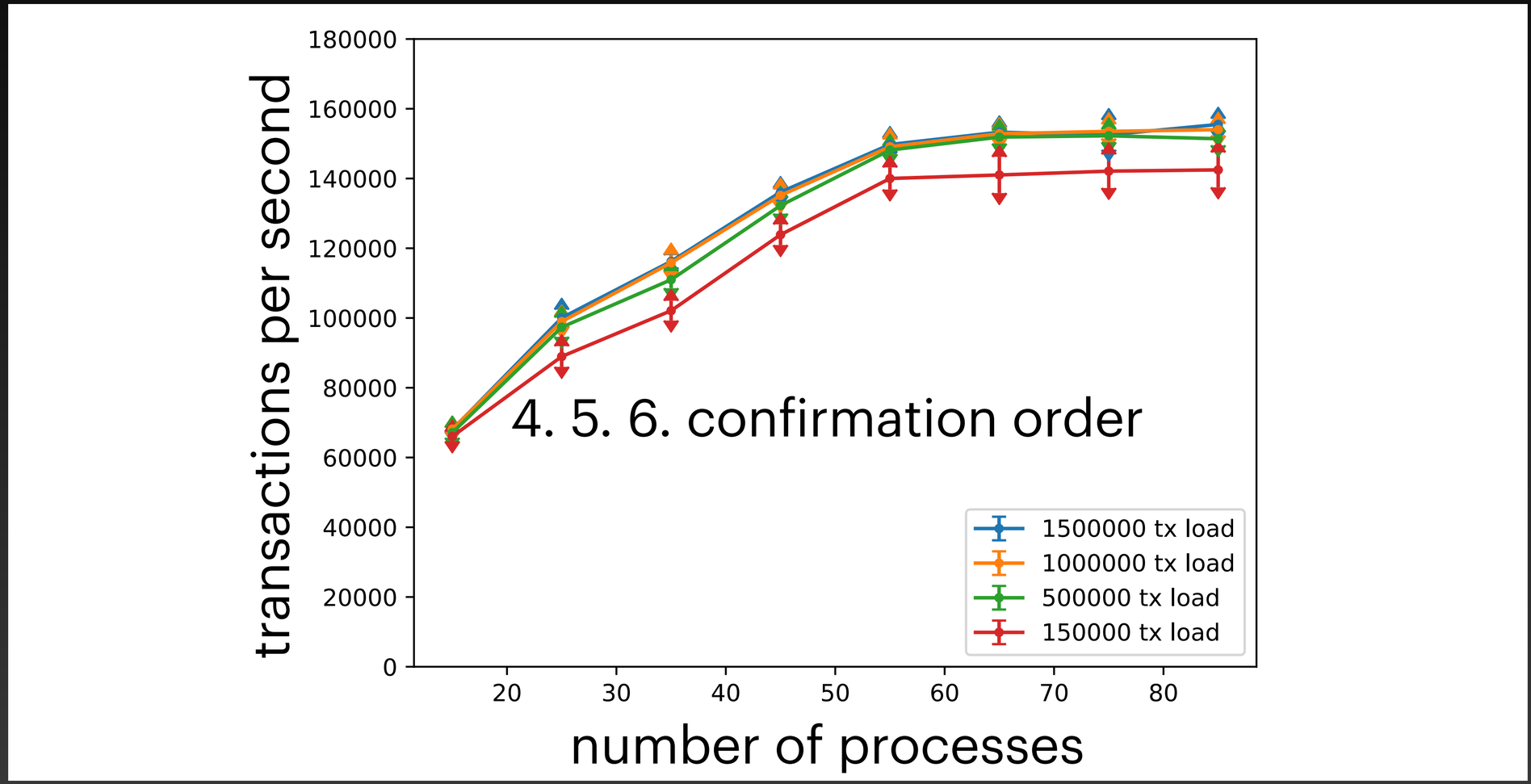
## High concurrency



# FastPay
## Robustness

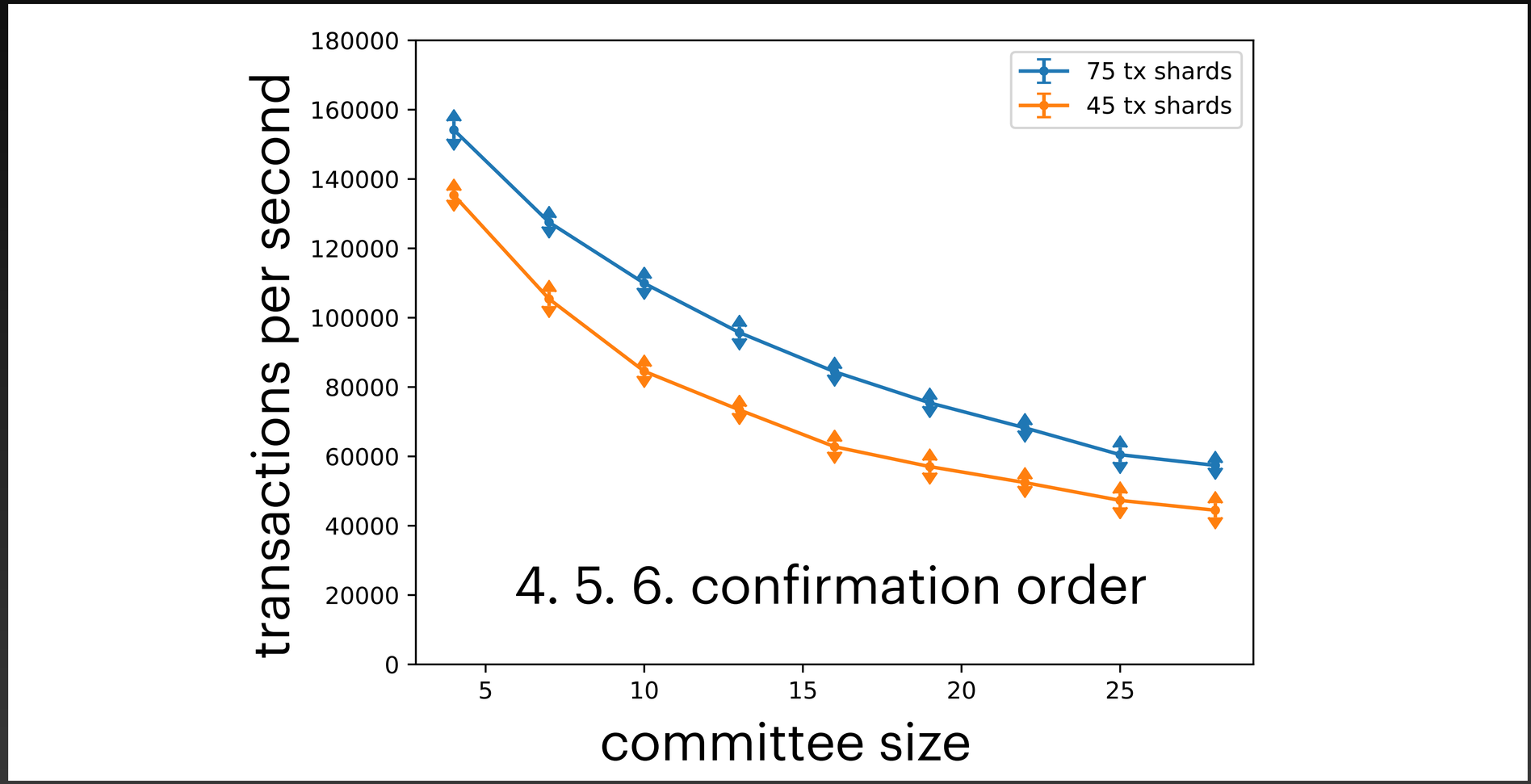


# FastPay
## Robustness



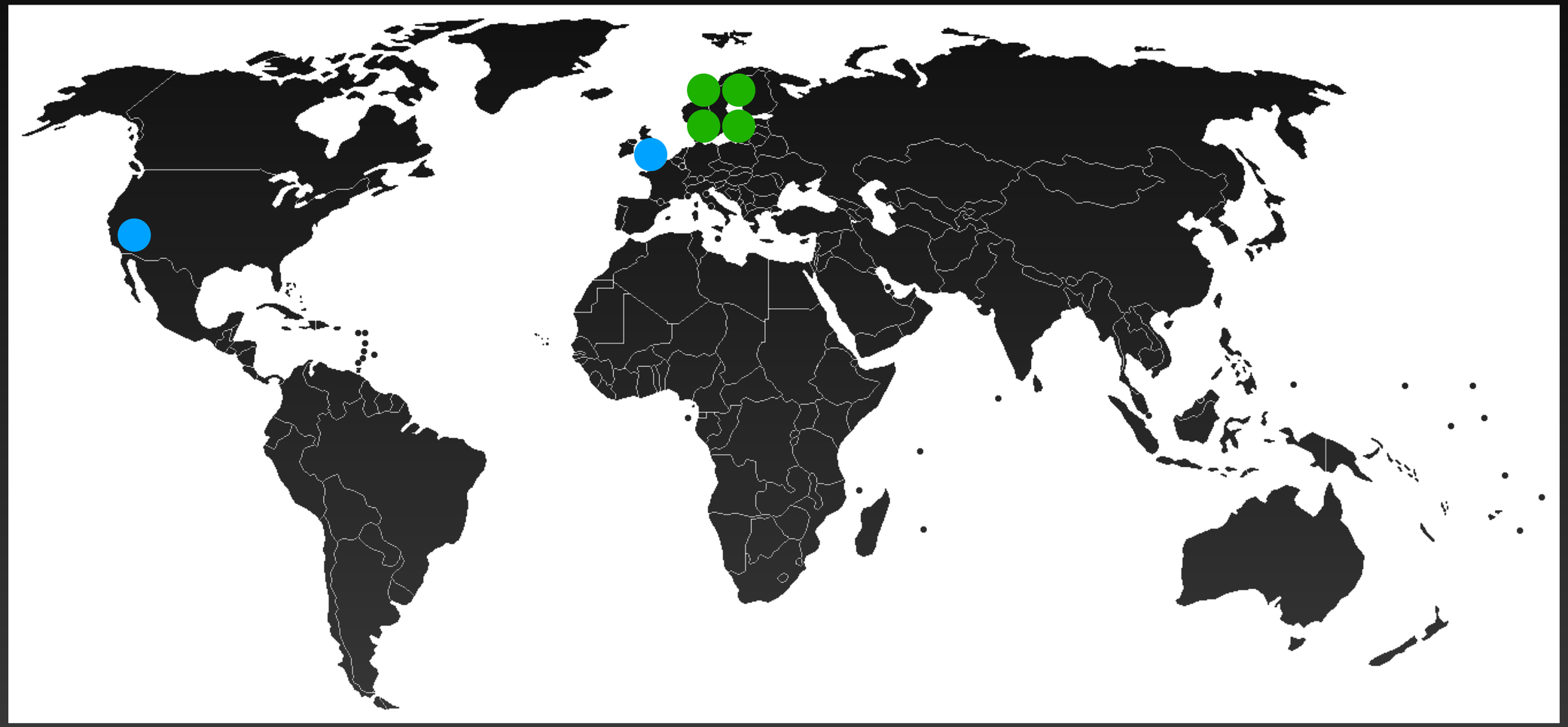
# FastPay

## Influence of the number of authorities

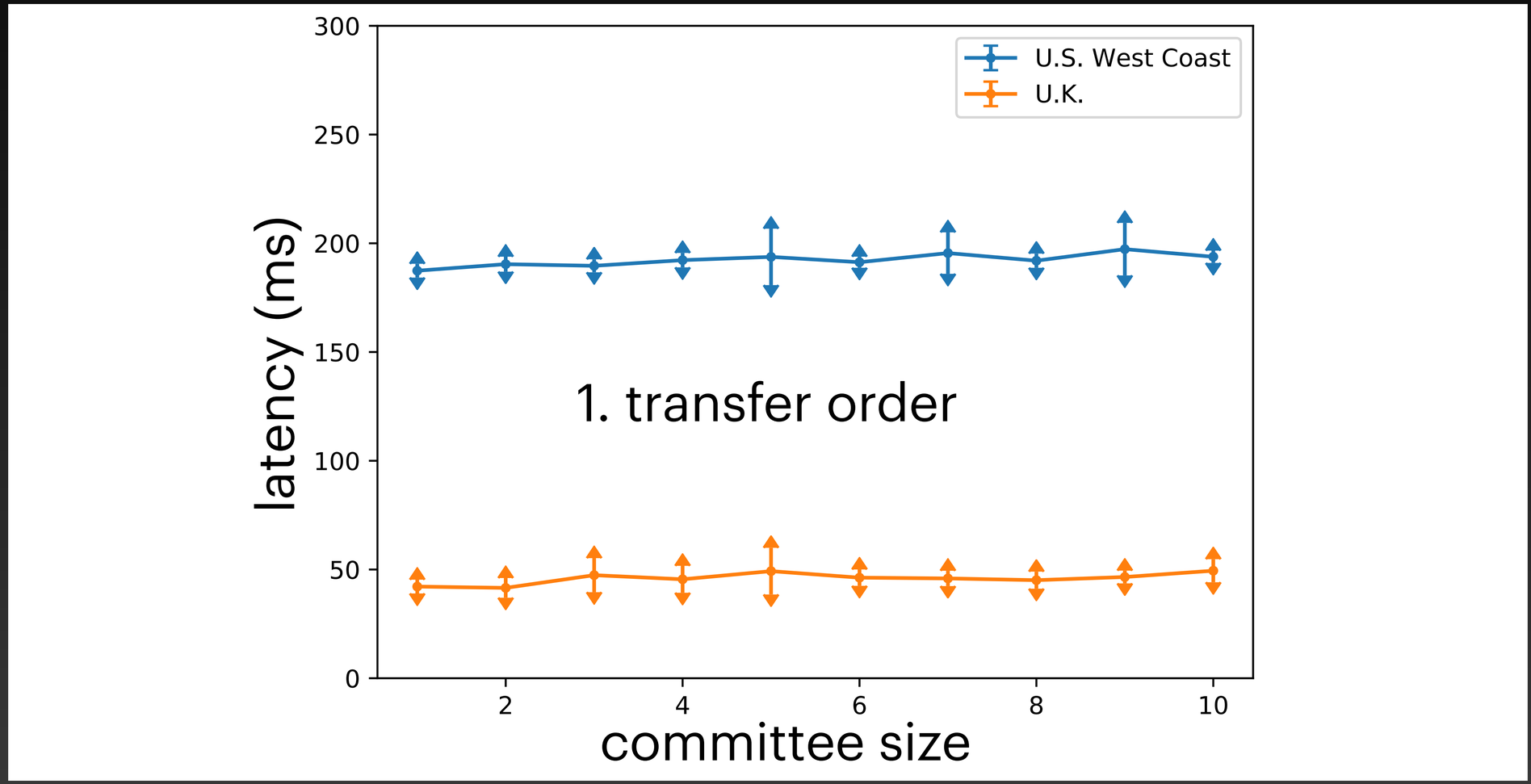


# FastPay

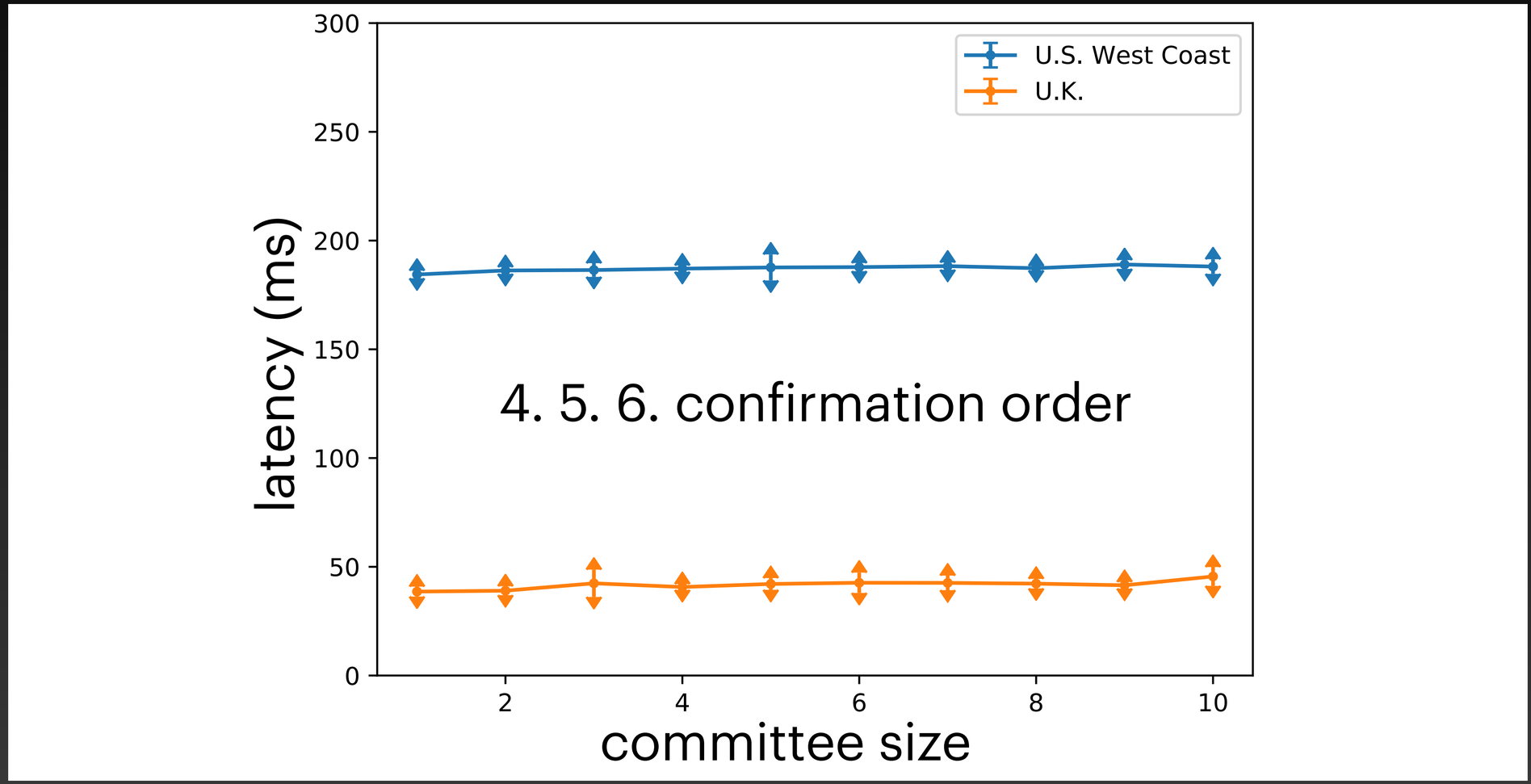
## Latency setup



# FastPay
## Latency

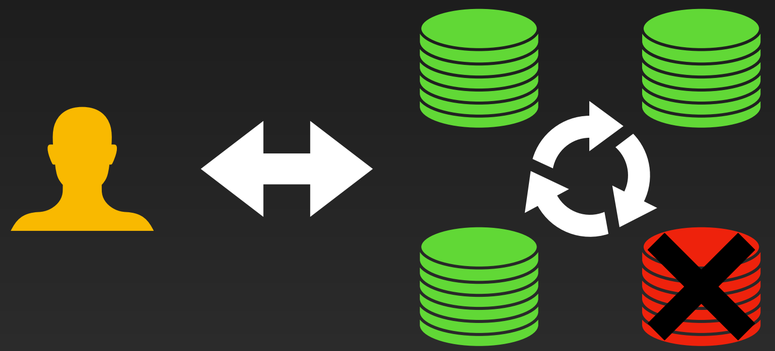


# FastPay
## Latency



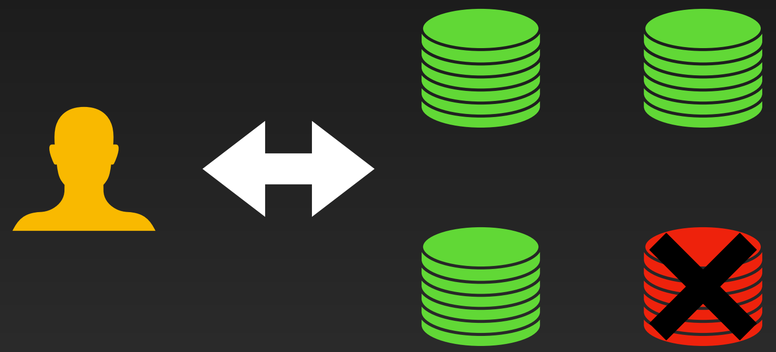
# Worst-case efficiency

## Blockchains



Bad leader can slow down the protocol

## FastPay



No leader, nothing changes

# FastPay

## The cost of simplicity

- Less than 4,000 LOC
- Over 1,500 Git commits
- Took 2.5 months to 3 engineers

# FastPay

## Deployment costs

- AWS m5d.8xlarge instance
- ~ 5 USD / hour

# FastPay
## Further works

- Checkpointing?
- Change the authorities?
- Privacy?

# Conclusion

## FastPay

- Based on Byzantine Consistent Broadcast
- Simple design, low latency, high capacity, very robust

- **Paper:** https://arxiv.org/abs/2003.11506
- **Code:** https://github.com/novifinancial/fastpay

# asonnino@fb.com

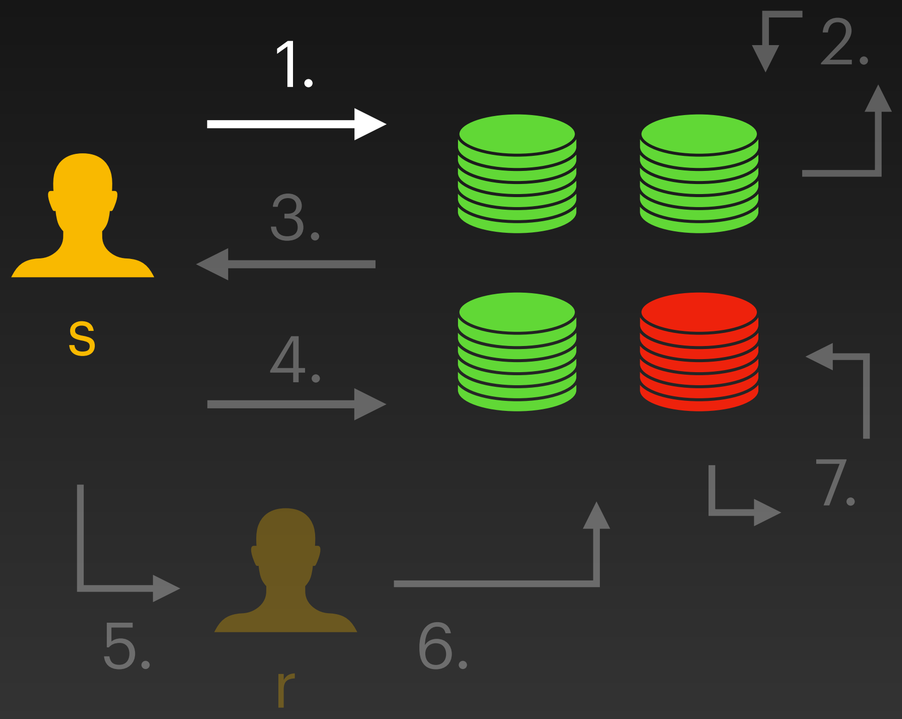
Alberto Sonnino

# EXTRA

# Protocol Details

From FastPay to FastPay

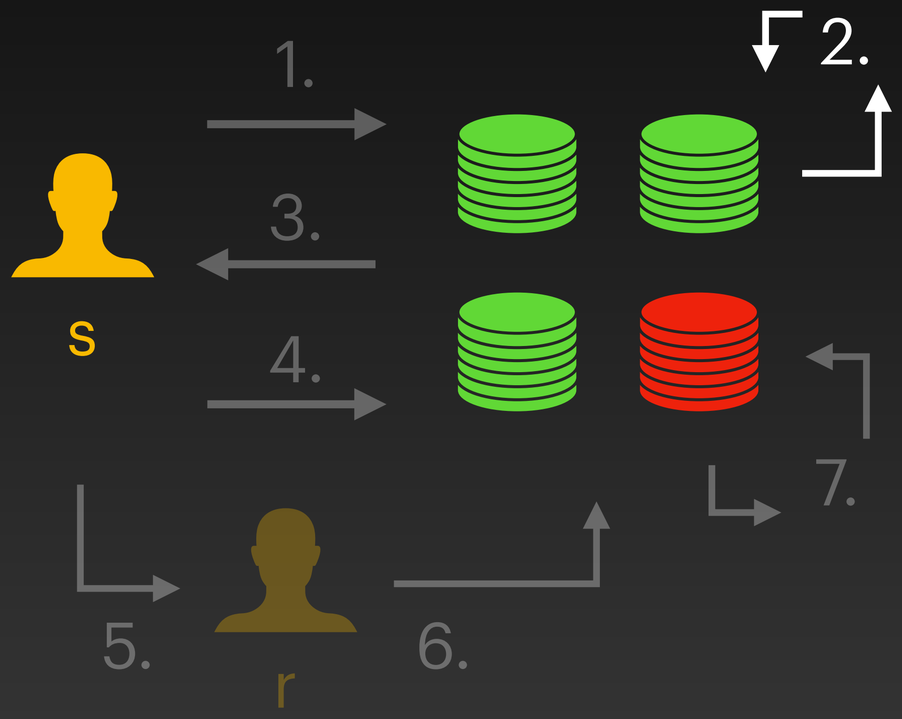
# FastPay
## Protocol details




### 1. transfer order

- Sender address
- Recipient address
- Amount
- Sequence number
- Sender's signature

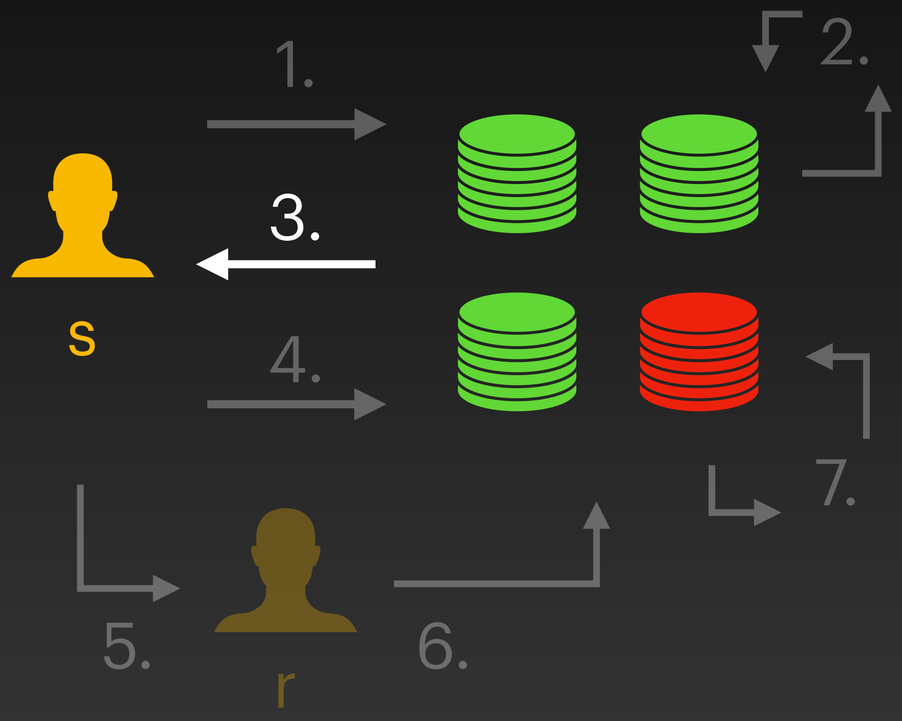
# FastPay
## Protocol details




### 2. verify

- The sender's signature
- No previous tx is pending
- The amount is positive
- Sequence number is as expected
- Balance is sufficient

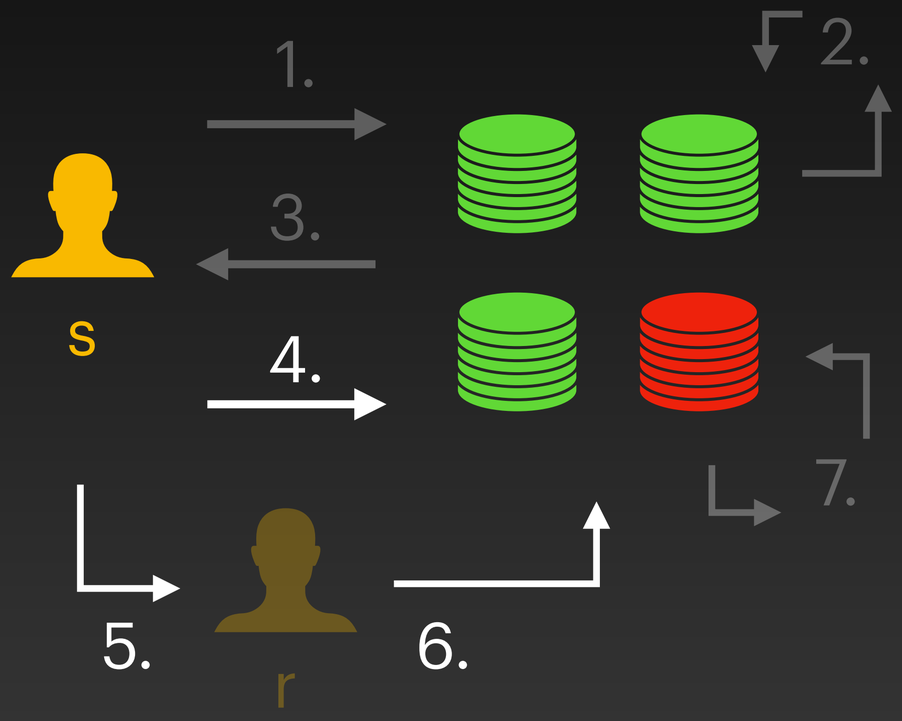
# FastPay
## Protocol details




### 3. signed transfer order

- Each authority signed the transfer order received in step 1.

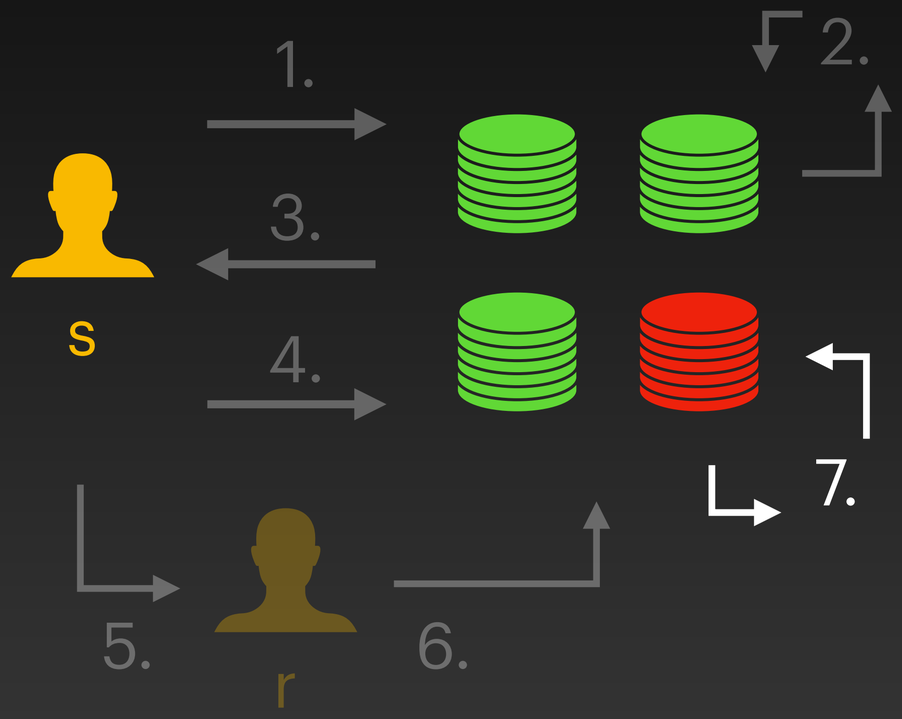
# FastPay
## Protocol details




## 4. 5. 6. confirmation order

- Collect enough signed transfer orders from step 2.

# FastPay
## Protocol details




### 7. update

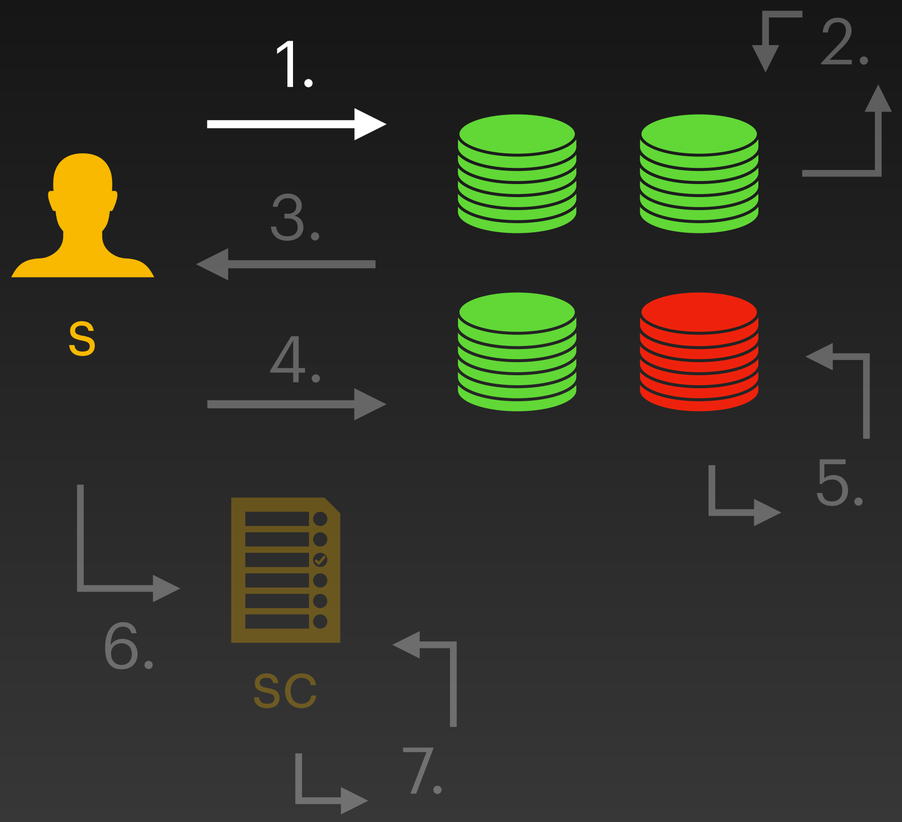
- Check there are enough signatures
- Decrease the senders' balance
- Increase the sequence number
- Set the pending order to None
- Increase the recipient's balance

# Protocol Details

From FastPay to primary infrastructure

# FastPay

## From FastPay to primary infrastructure

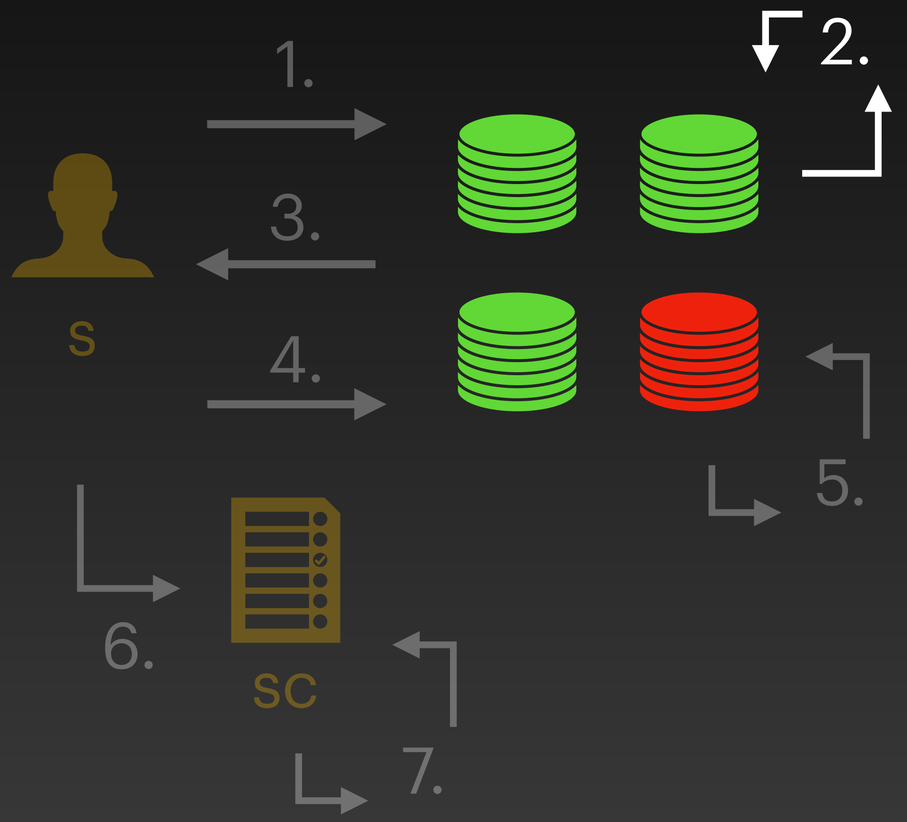



## 1. transfer order

- Sender address
- Recipient address
- Amount
- Sequence number
- Sender's signature

# FastPay

## From FastPay to primary infrastructure

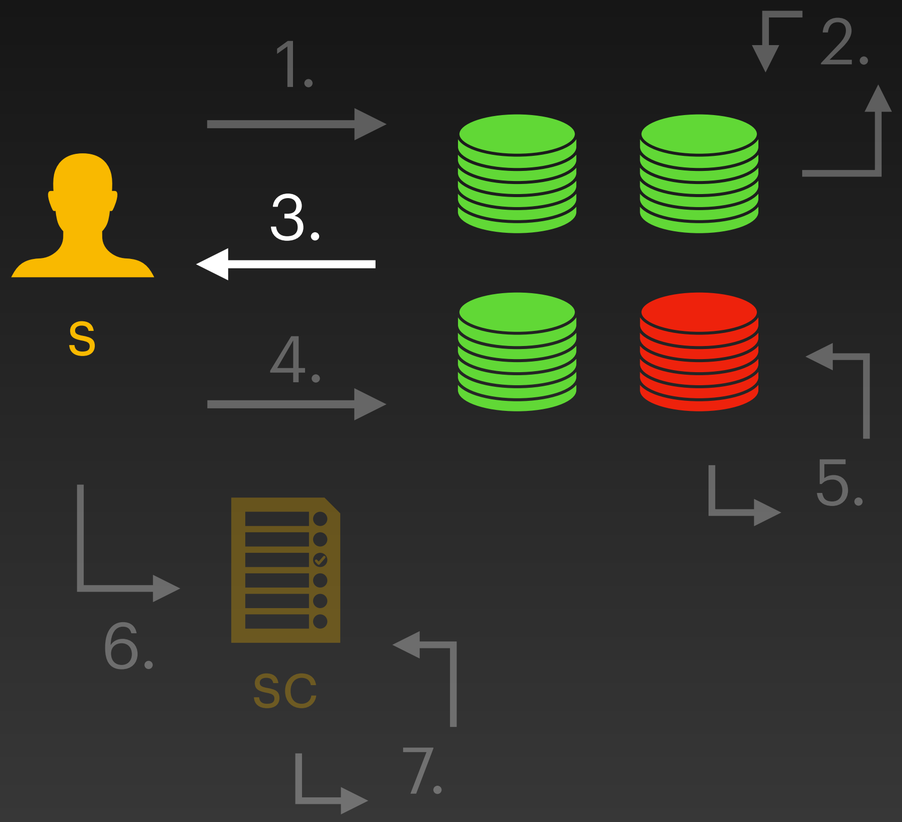



## 2. verify

- The sender's signature
- No previous tx is pending
- The amount is positive
- Sequence number is as expected
- Balance is sufficient

# FastPay

## From FastPay to primary infrastructure

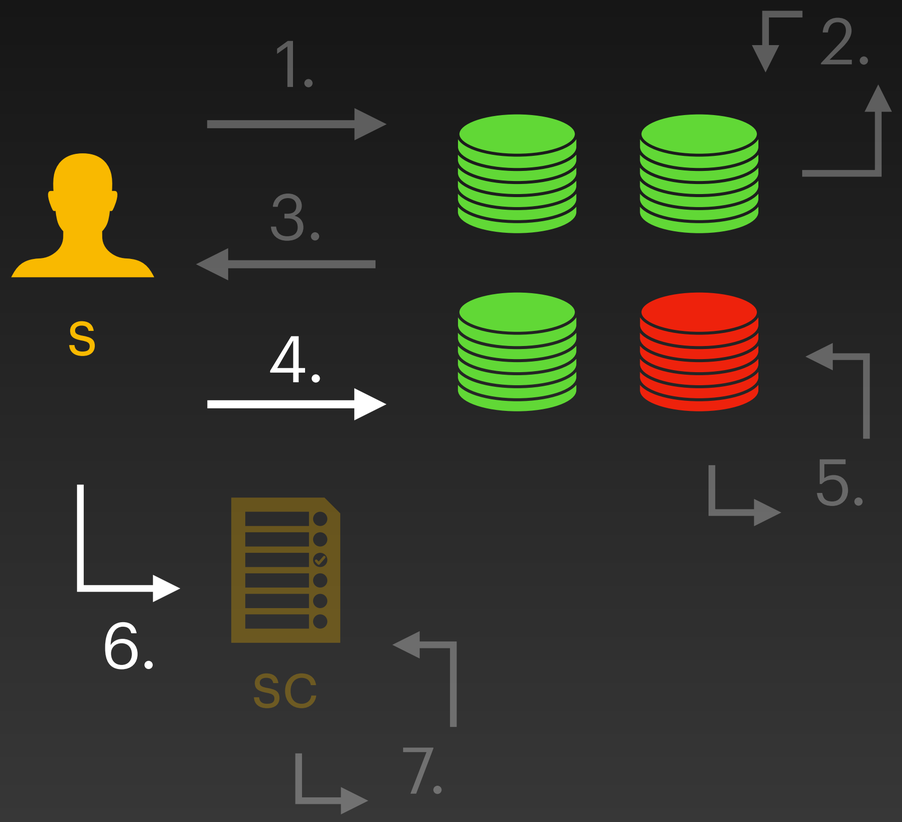



### 3. signed transfer order

- Each authority signed the transfer order received in step 1.

# FastPay

## From FastPay to primary infrastructure

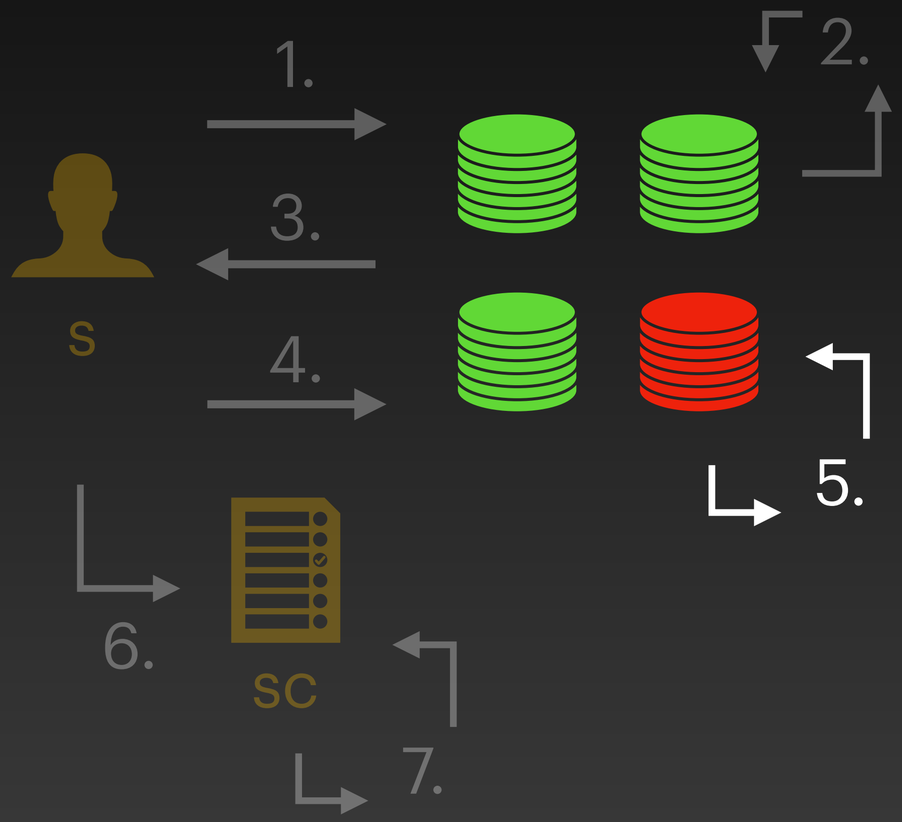



### 4. 6. confirmation order

- Collect enough signed transfer orders from step 2.

# FastPay

## From FastPay to primary infrastructure

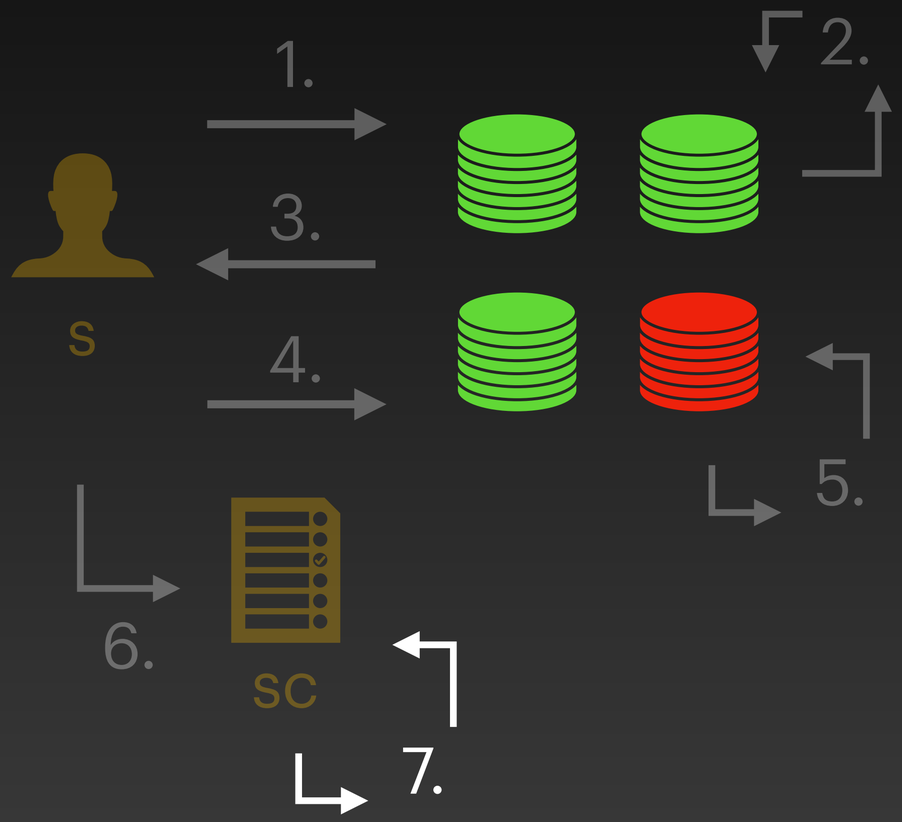



## 5. update

- Check there are enough signatures
- Decrease the senders' balance
- Increase the sequence number
- Set the pending order to None

# FastPay

## From FastPay to primary infrastructure




## 7. verify & update

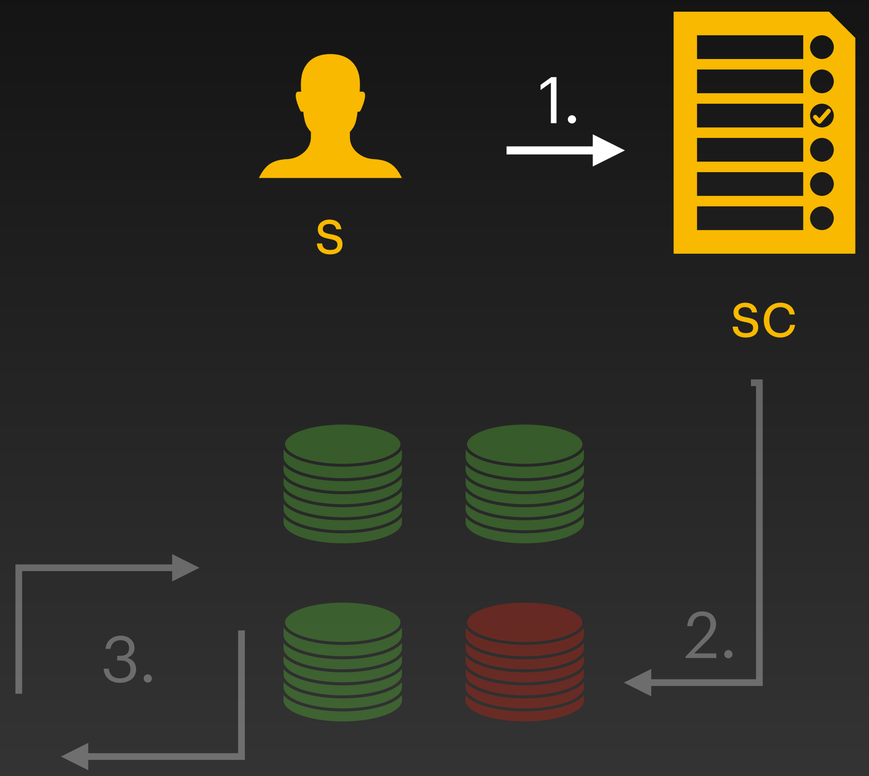
- Check sequence number is not on the redeem log
- Update the redeem log
- Transfer the amount to recipient

# Protocol Details

From primary infrastructure to FastPay

# FastPay

## From primary infrastructure to FastPay

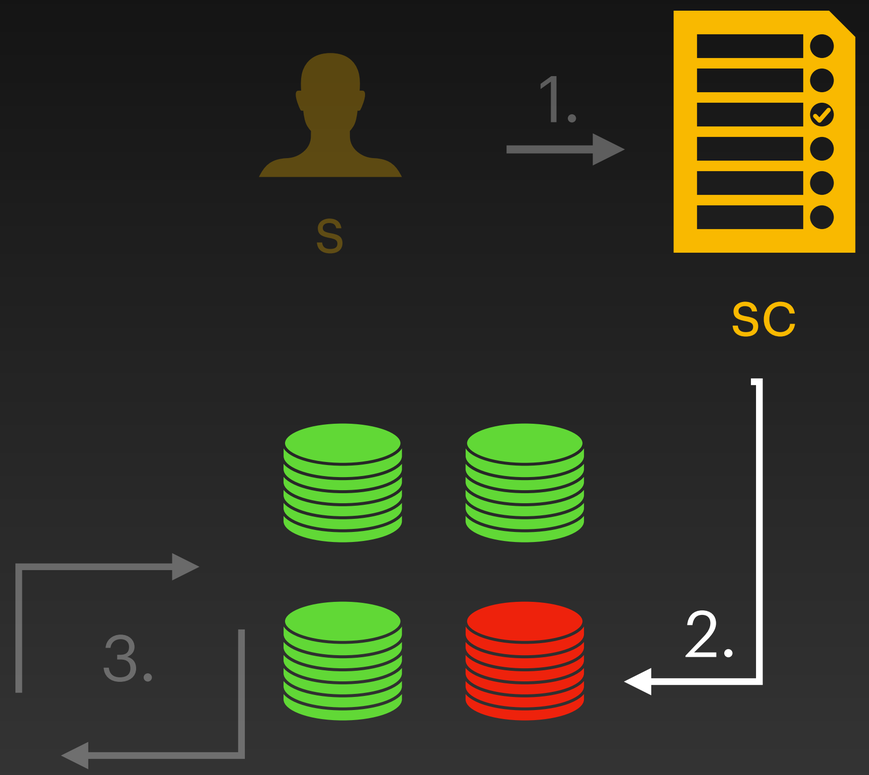



### 1. funding transaction

- FastPay recipient
- All fields required by the primary infrastructure (and the amount)

# FastPay

## From primary infrastructure to FastPay

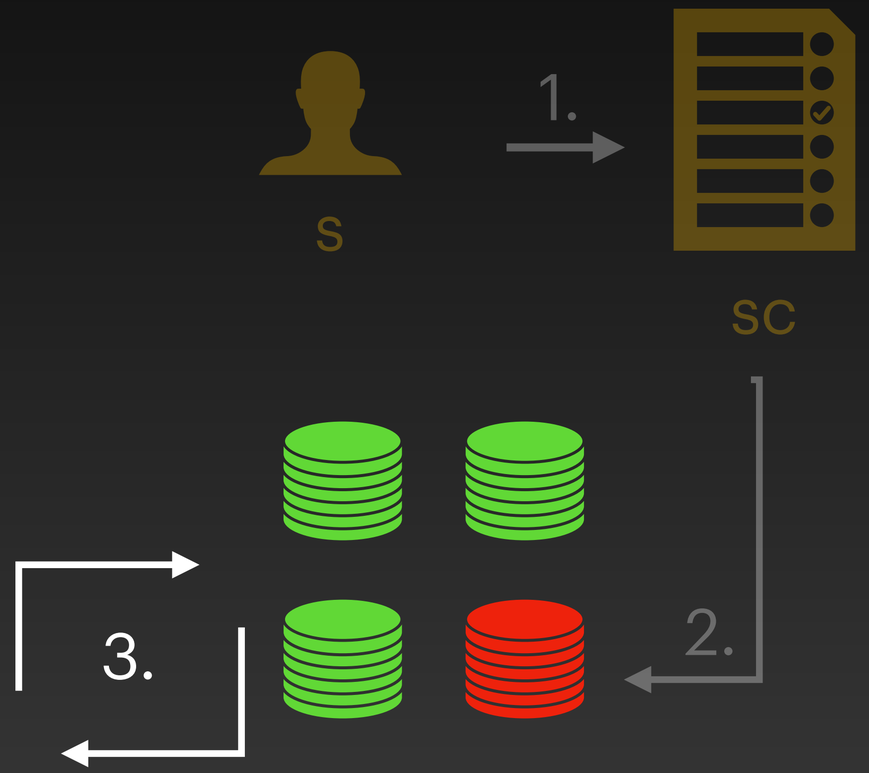



## 2. synchronization order

- Read the transaction on the primary infrastructure (once it is sequenced)

# FastPay

## From primary infrastructure to FastPay




### 3. update & verify

- Check last primary tx index
- Increment last primary tx index
- Create a FastPay account for the recipient (if needed)
- Increase recipient's balance