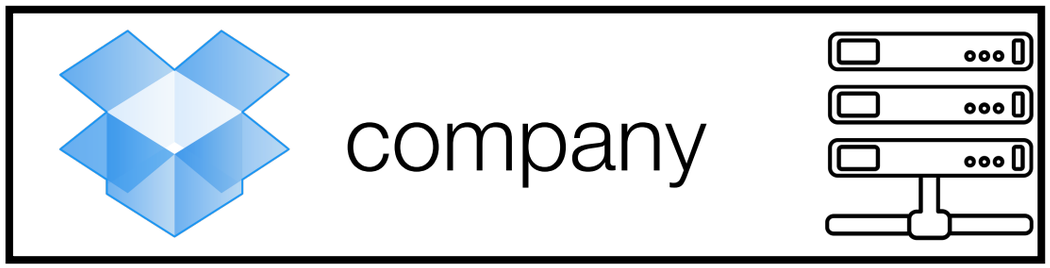
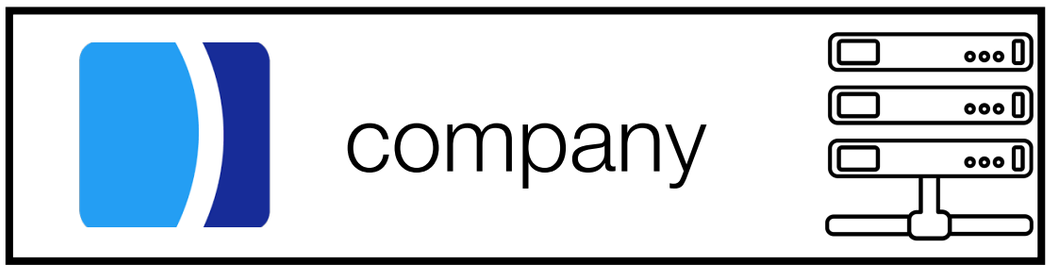
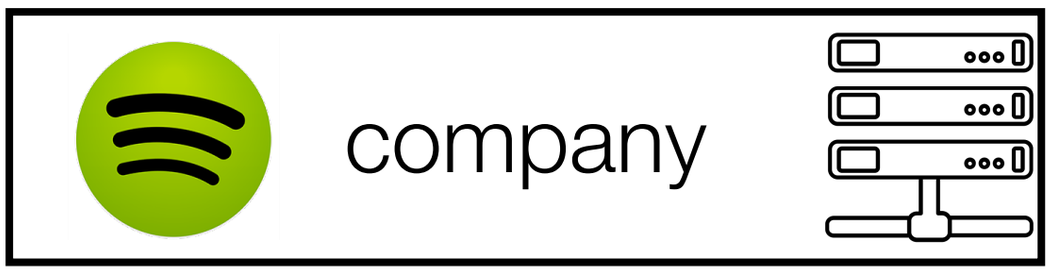
Alternative architectures for distributed ledgers

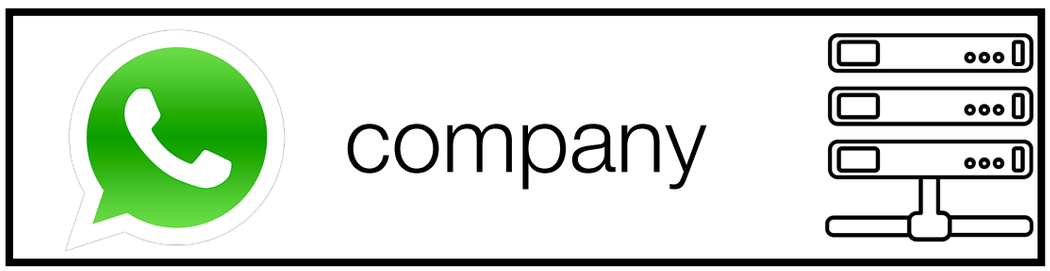
Sarah Meiklejohn (University College London)

data consumers





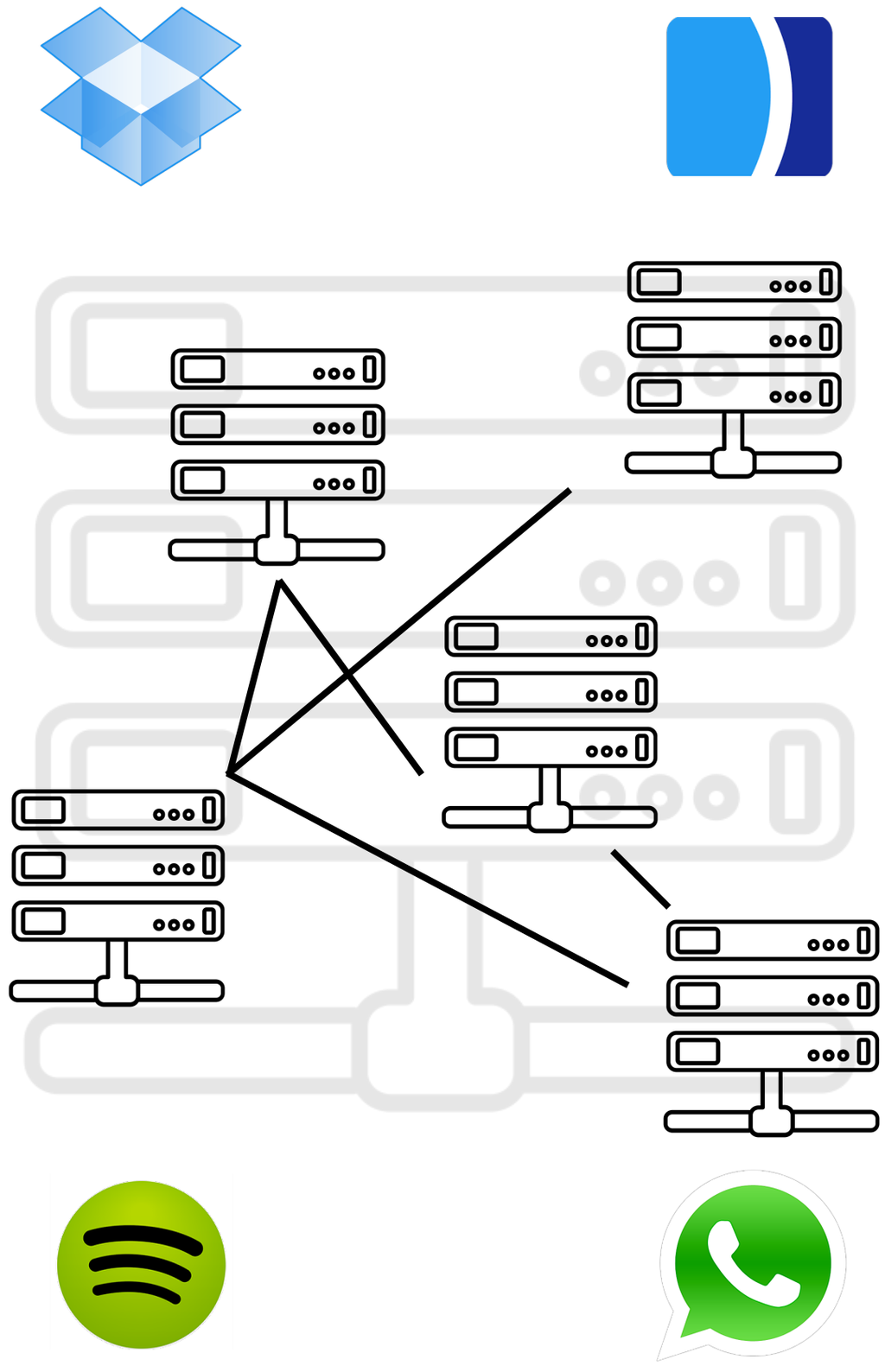




data producers

(icons by parkjisun from noun project)

data consumers



data producers

(icons by parkjisun from noun project)

top ten obstacles for blockchains

10 usability

9 governance

8 comparisons

7 key management

6 agility

5 interoperability

4 scalability

3 cost-effectiveness

2 privacy

1 scalability

10 usability

9 governance

8 comparisons

7 key management

6 agility

5 interoperability

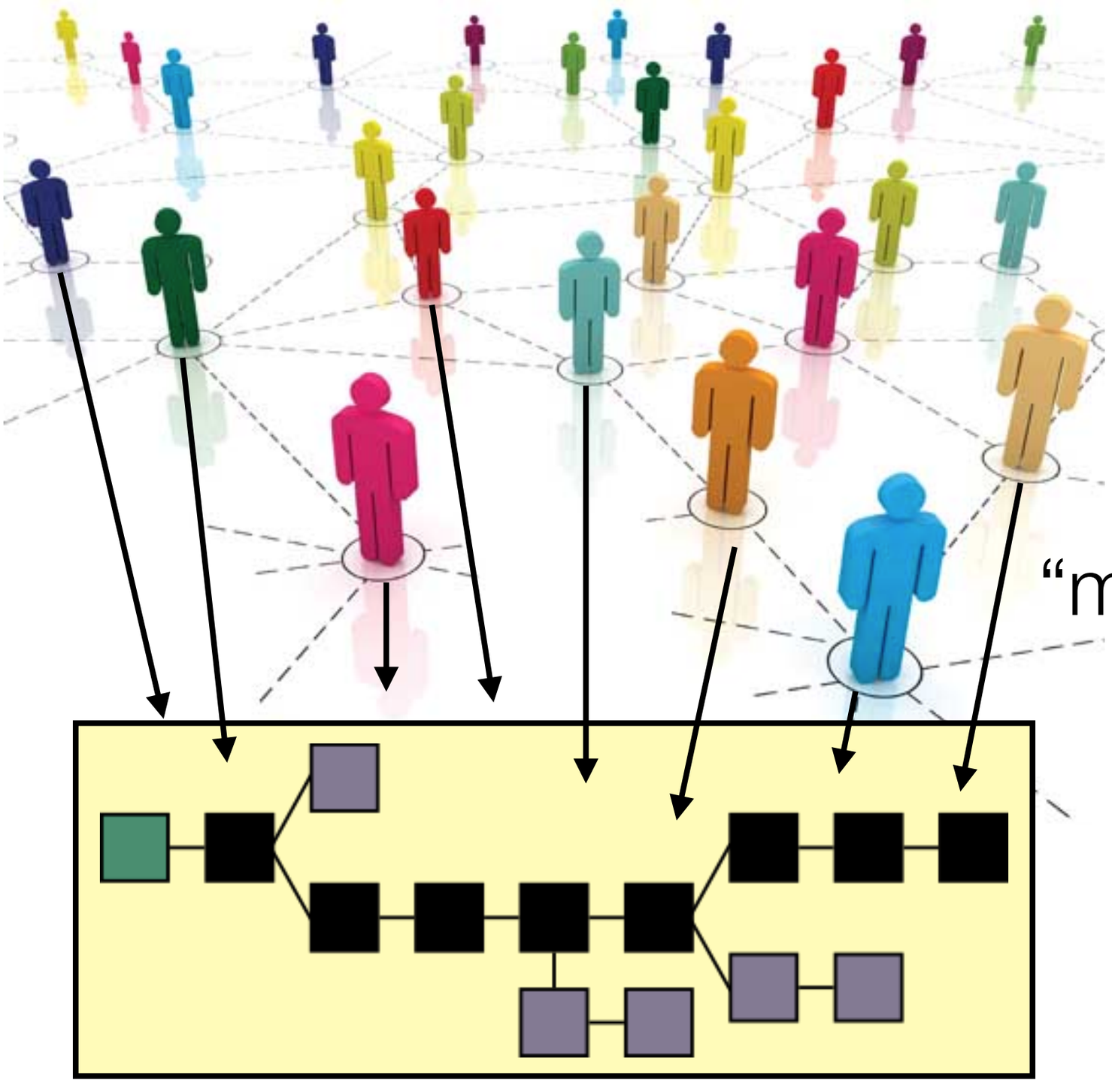
4 scalability

3 cost-effectiveness

2 privacy

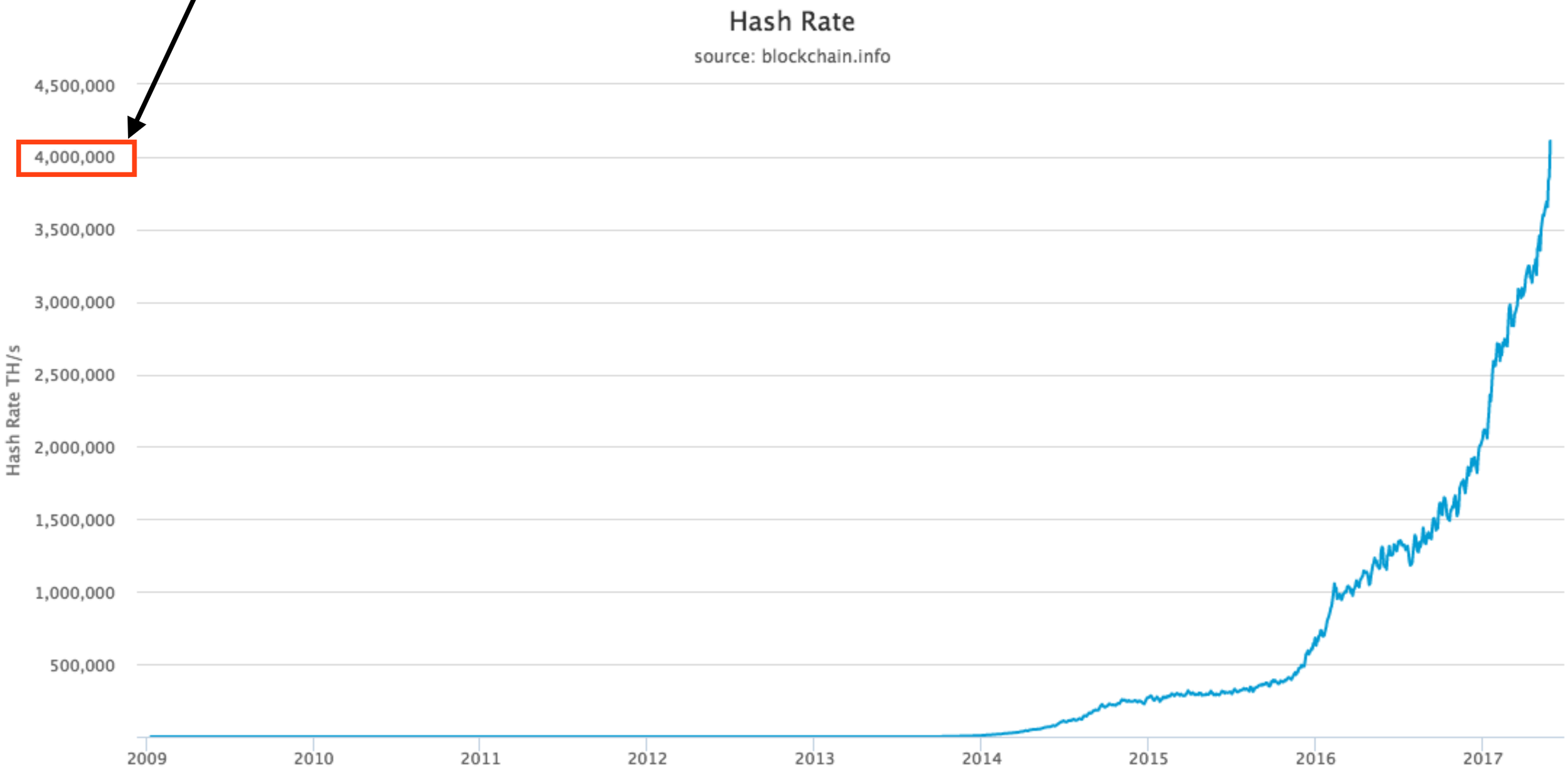
1 scalability

Bitcoin / blockchains / distributed ledgers



“mining”

over 4 EH/s (4×10^{18} H/s) to achieve 7 tx/s!



10 usability

9 governance

8 comparisons

7 key management

6 agility

5 interoperability

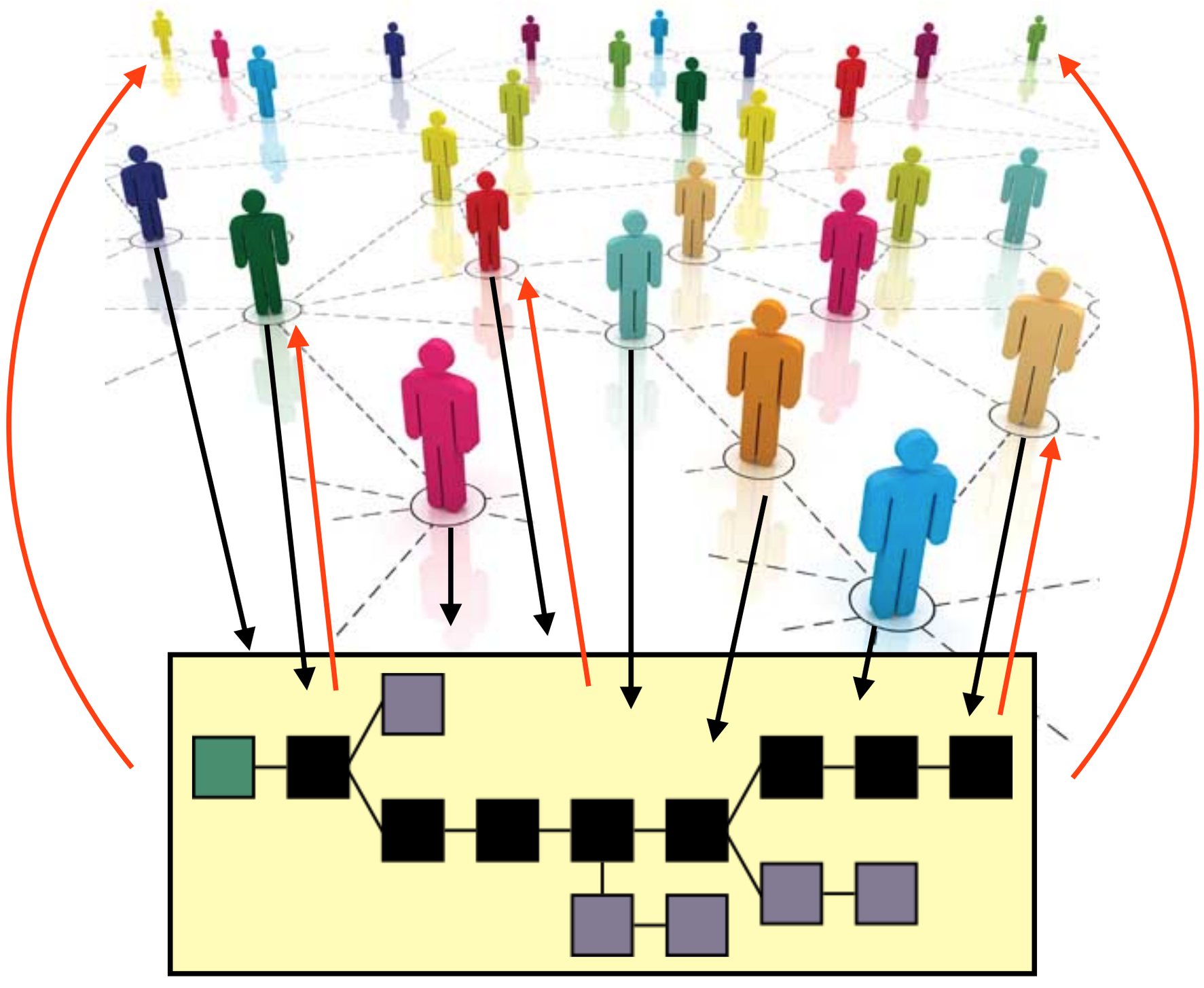
4 scalability

3 cost-effectiveness

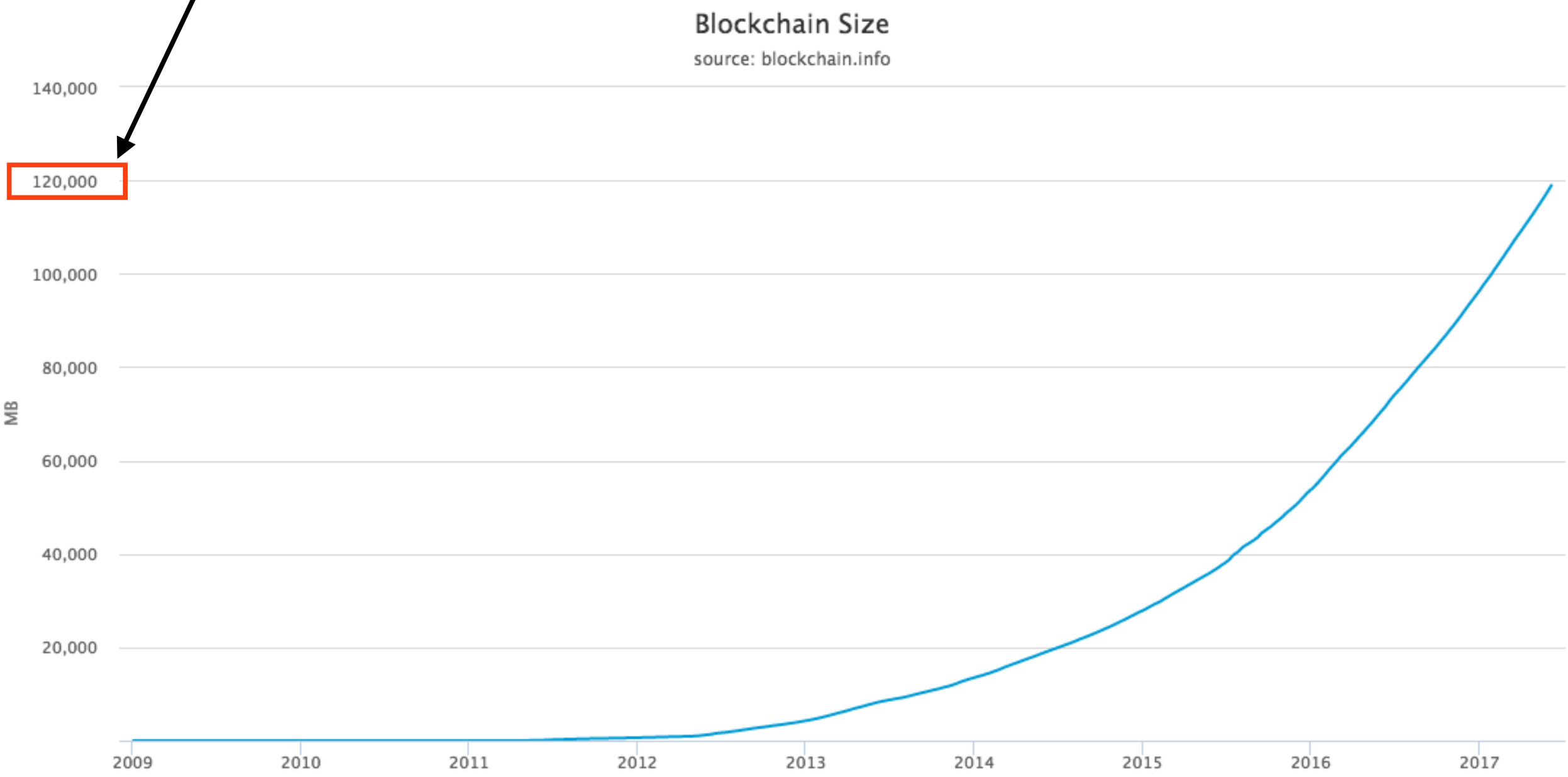
2 privacy

1 scalability

full state replication



120 GB and (always) rising



10 usability

9 governance

8 comparisons

7 key management

6 agility

5 interoperability

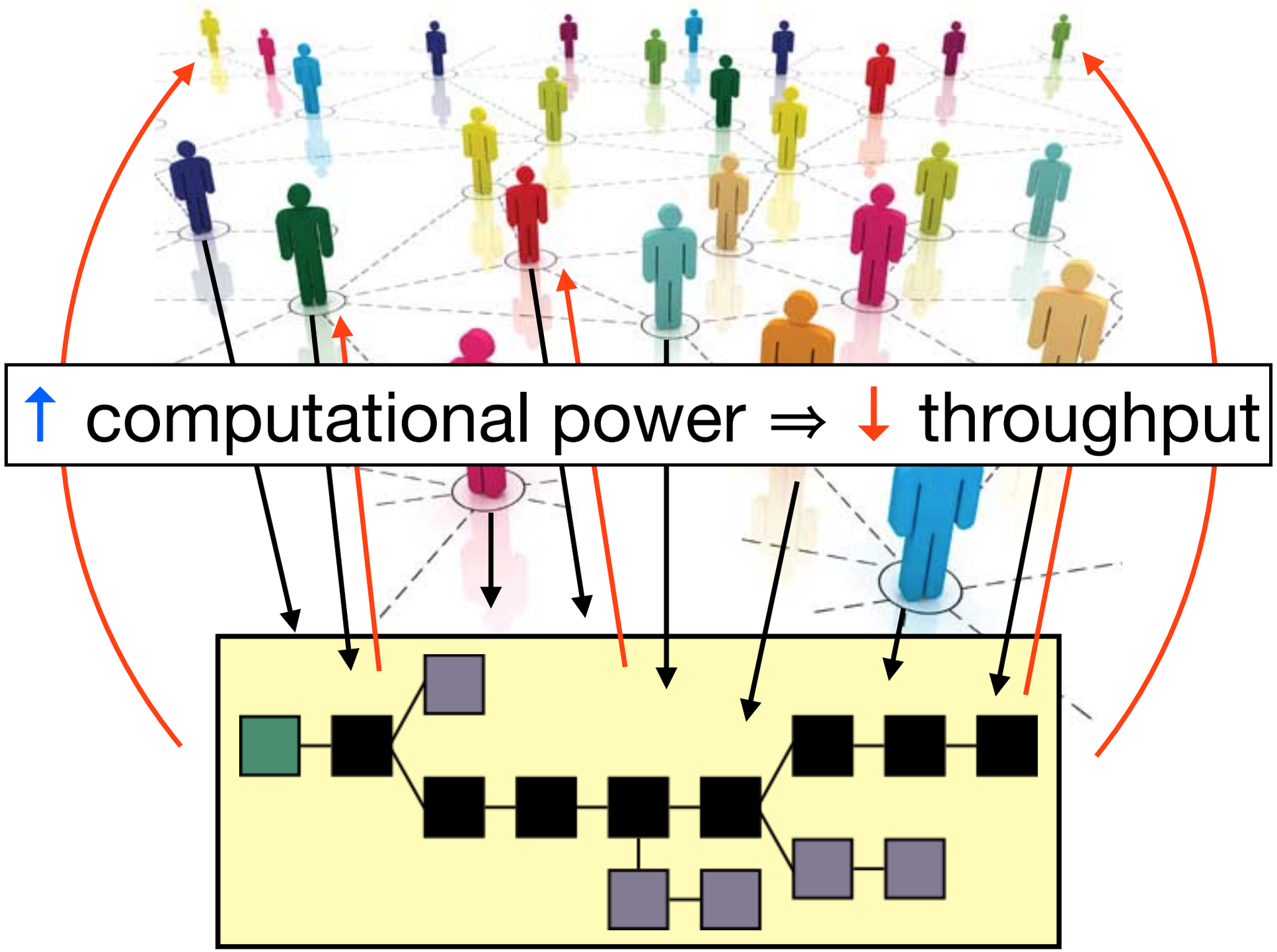
4 scalability

3 cost-effectiveness

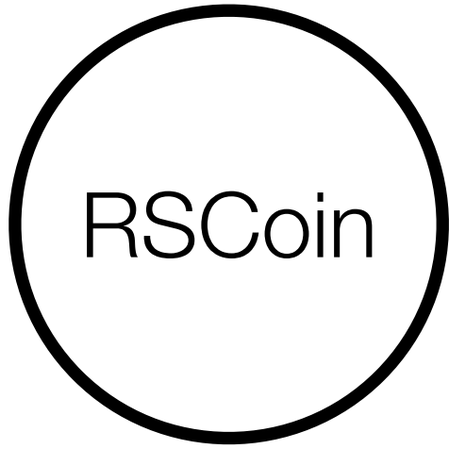
2 privacy

1 scalability

full state replication

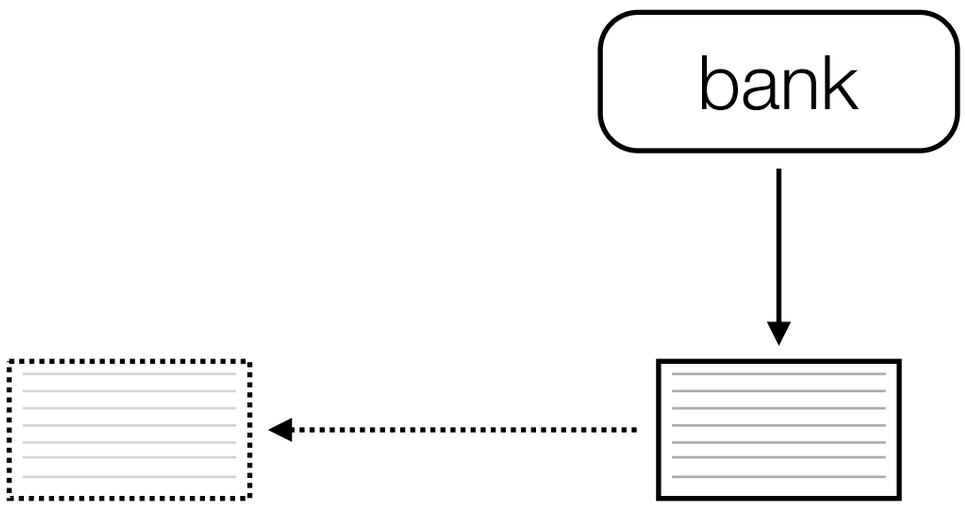


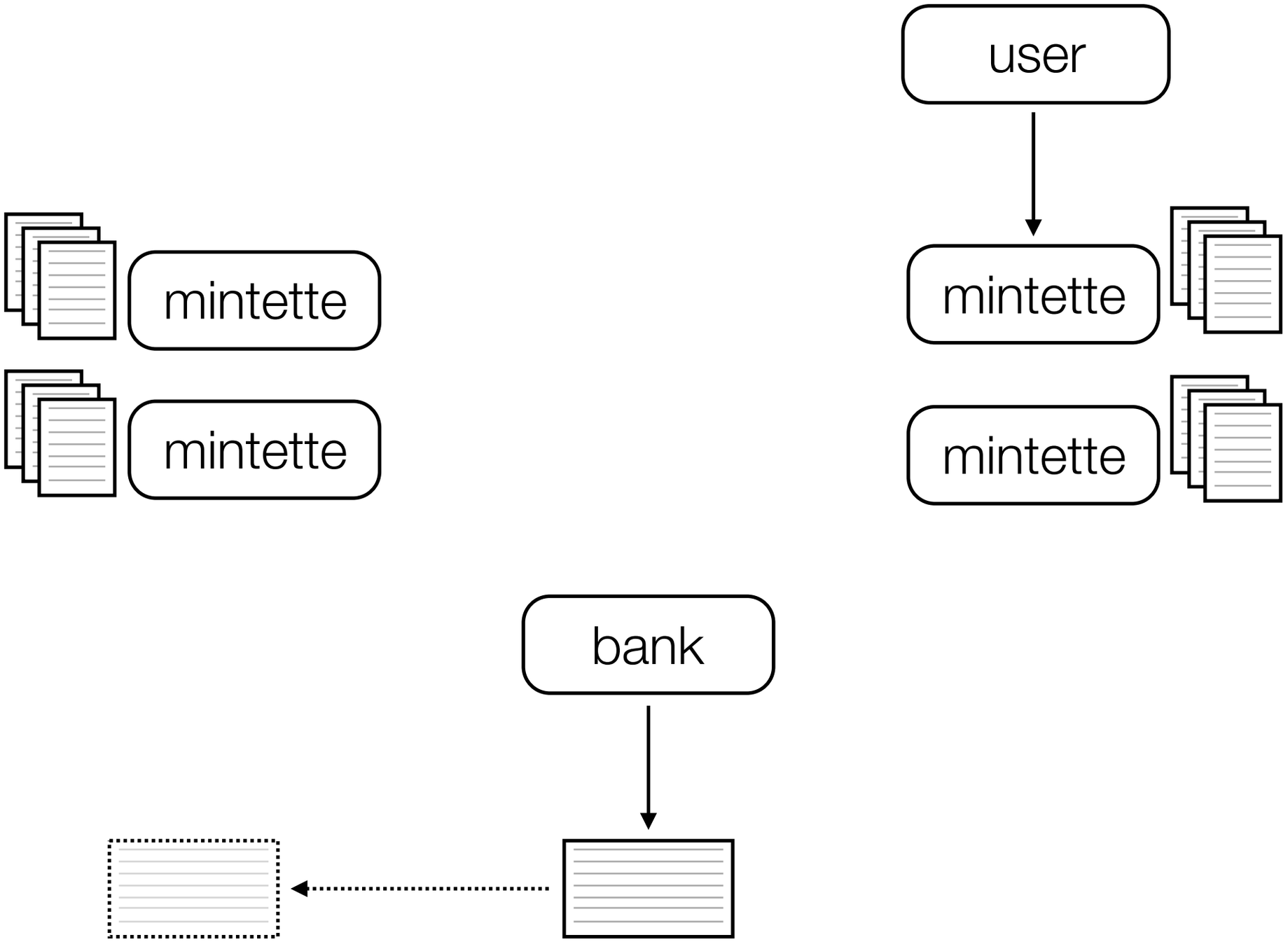
RSCoin [D**M** NDSS'16]



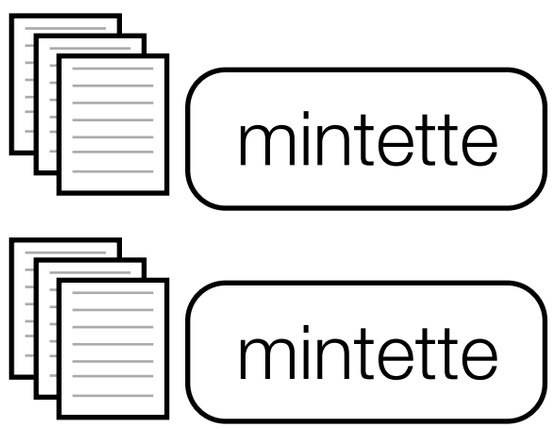
monetary supply	decentralized	centralized	centralized
ledger	decentralized	distributed	centralized
transparent?	y	y (or n)	n
pseudonyms?	y	y (or n)	n
computation	high!	low	low

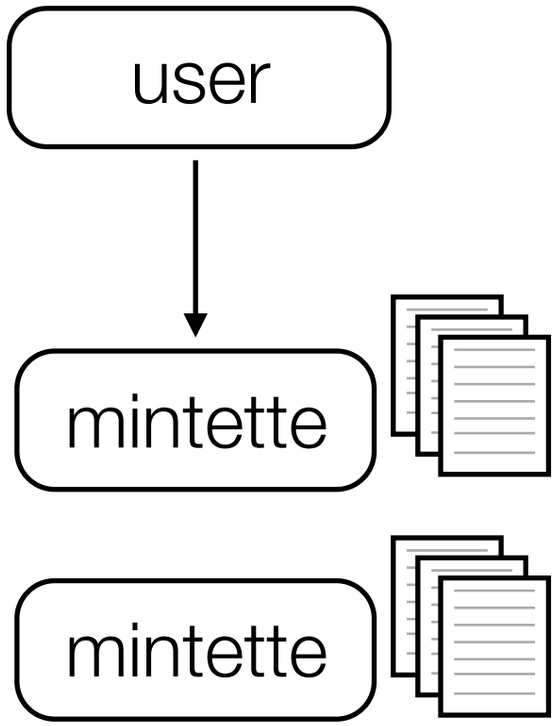
user

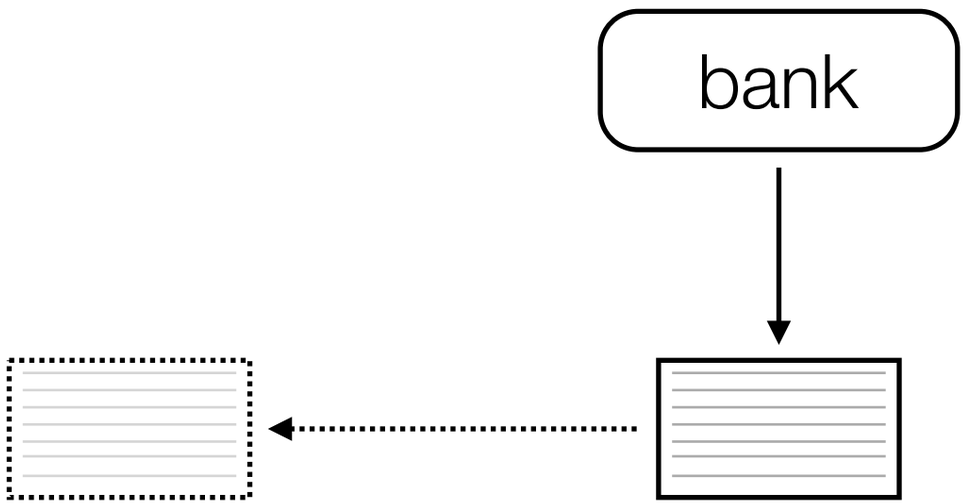


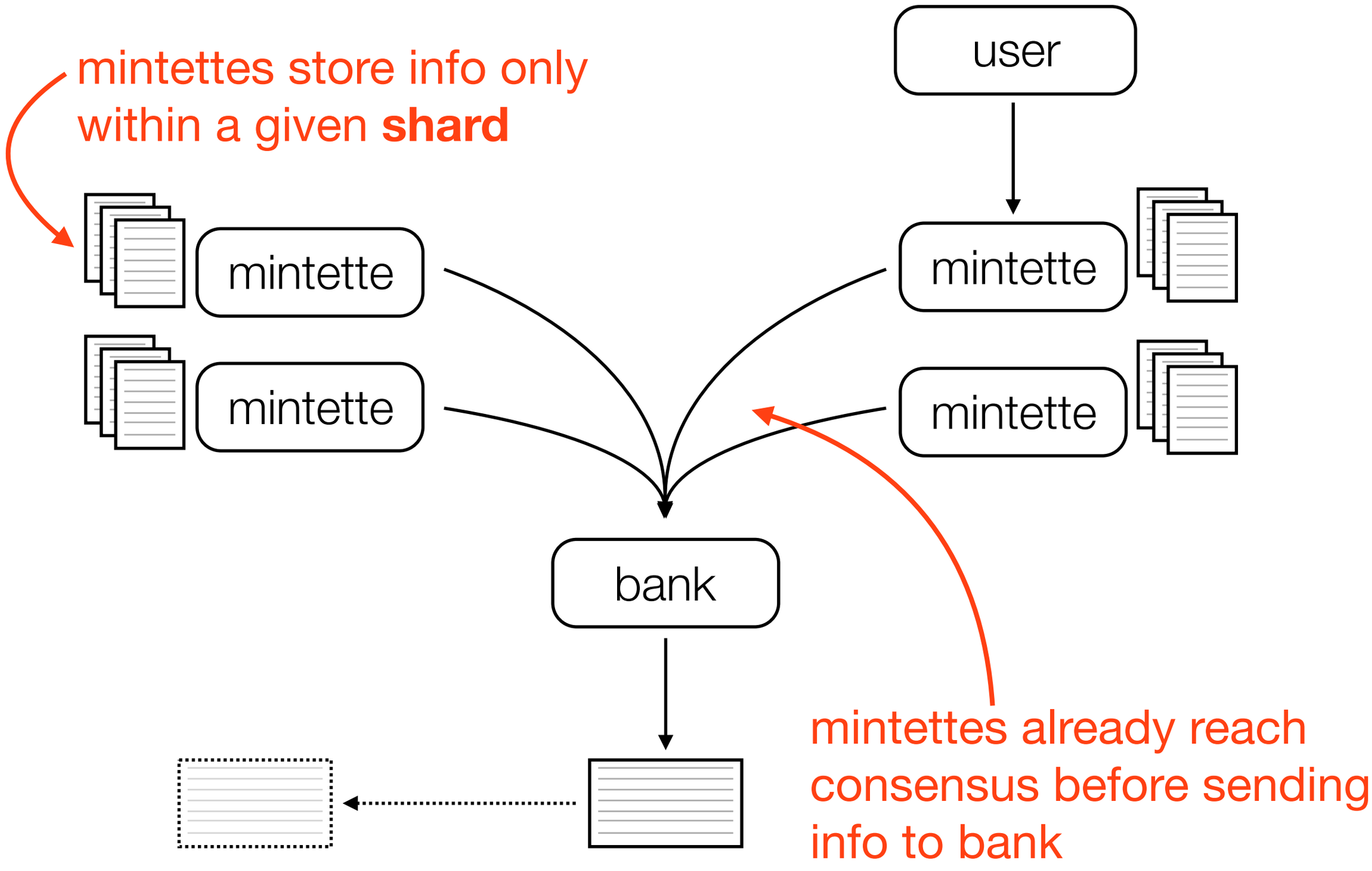

mintette
mintette
user
mintette
mintette
bank

mintettes store info only within a given **shard**

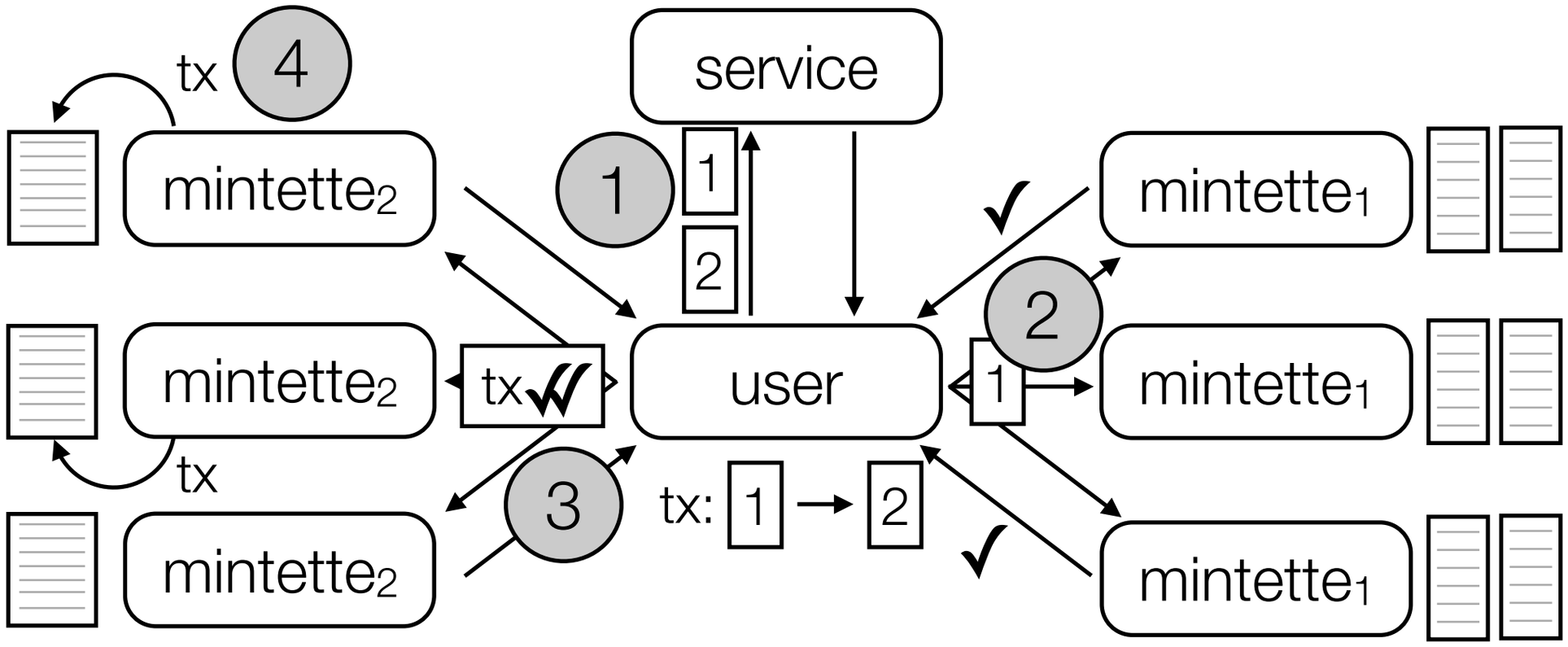




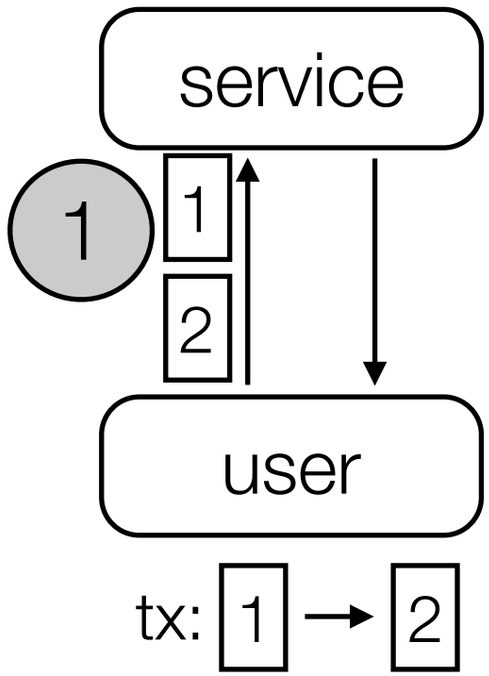


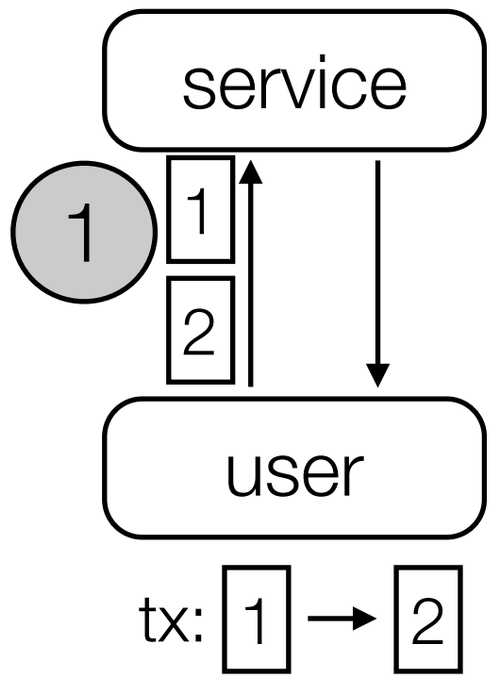

mintettes store info only
within a given **shard**
user
mintette
mintette
mintette
mintette
bank
mintettes already reach
consensus before sending
info to bank

RSCoin consensus

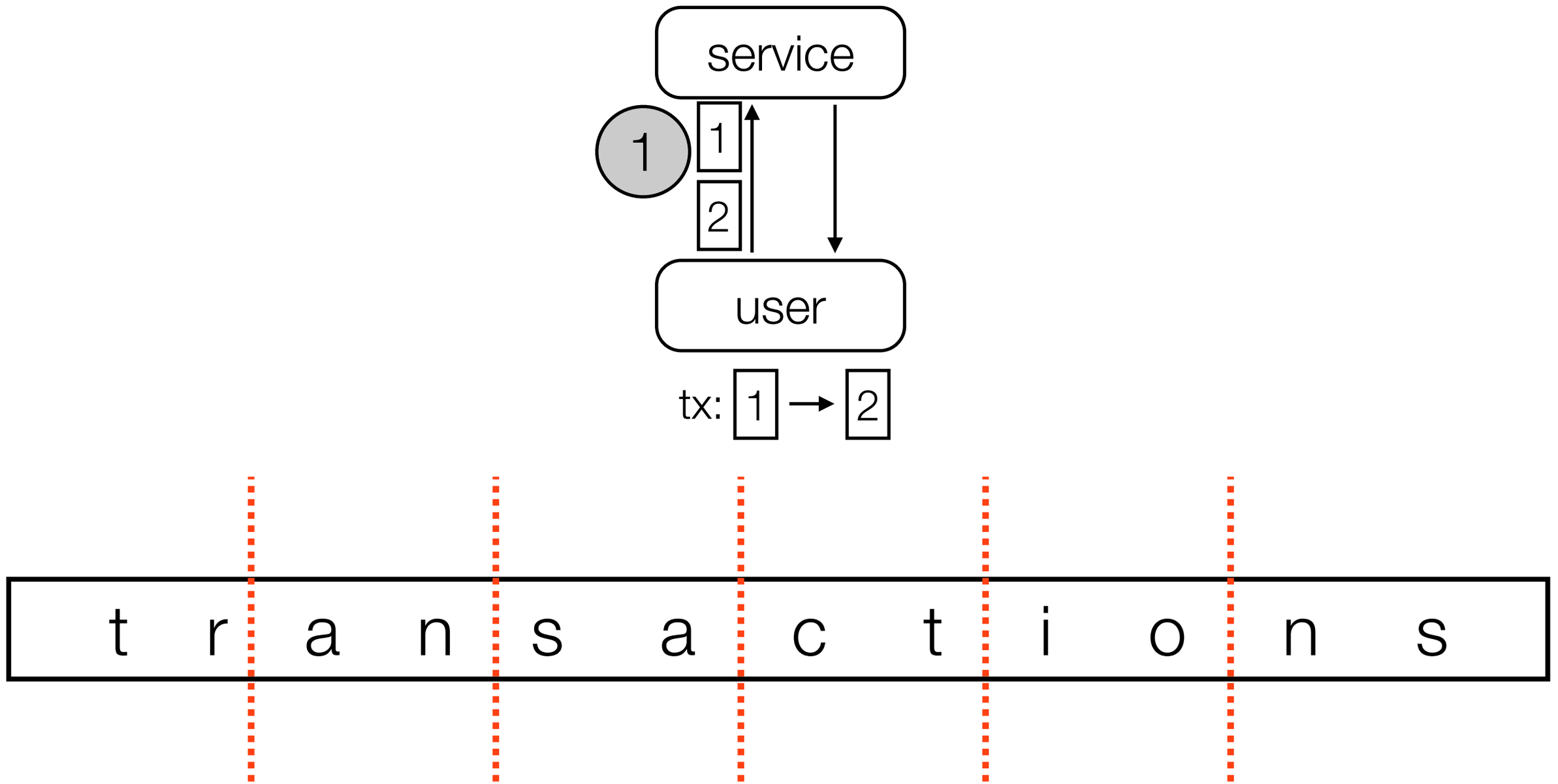


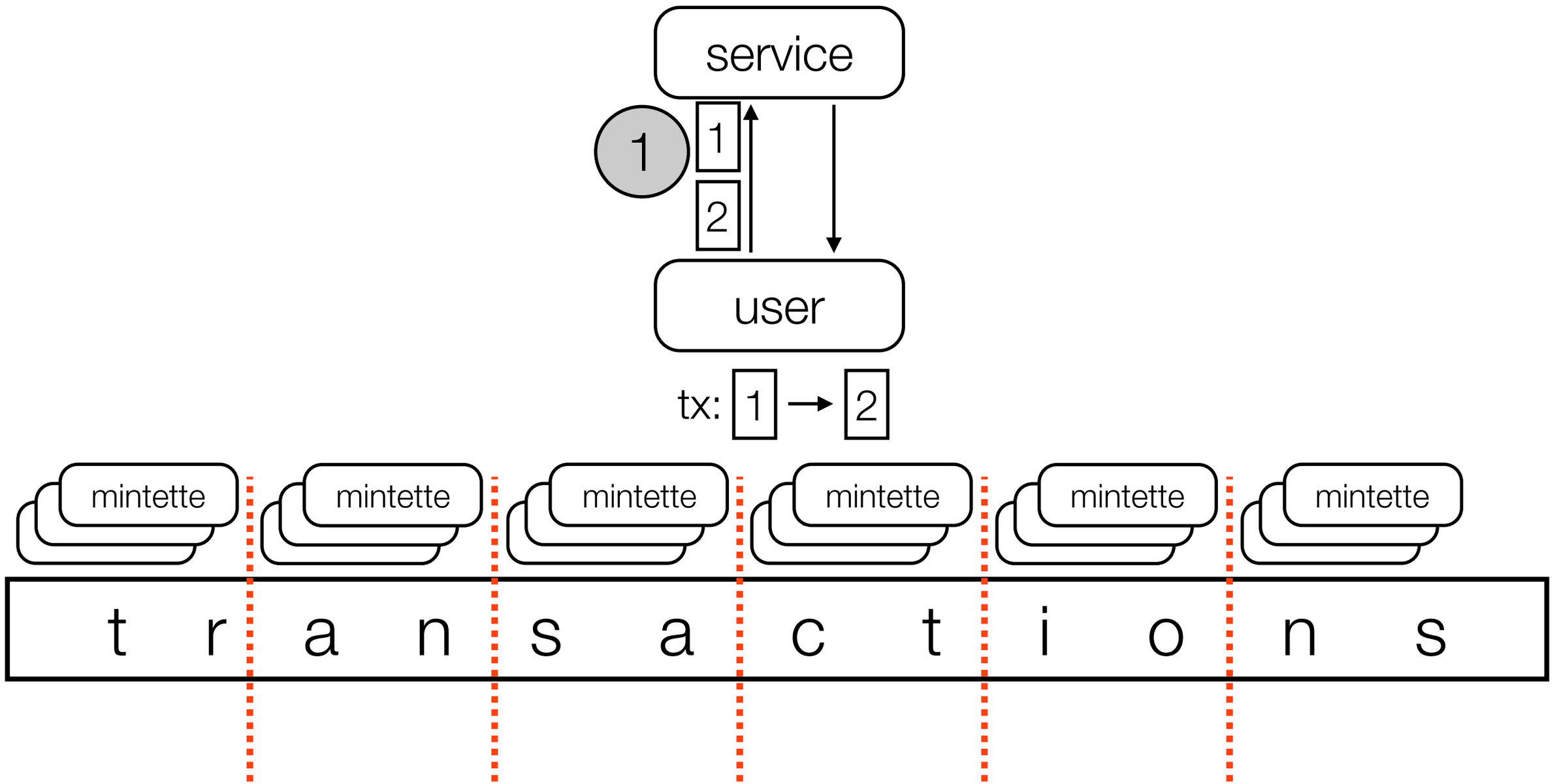
simple adaptation of Two-Phase Commit (2PC)

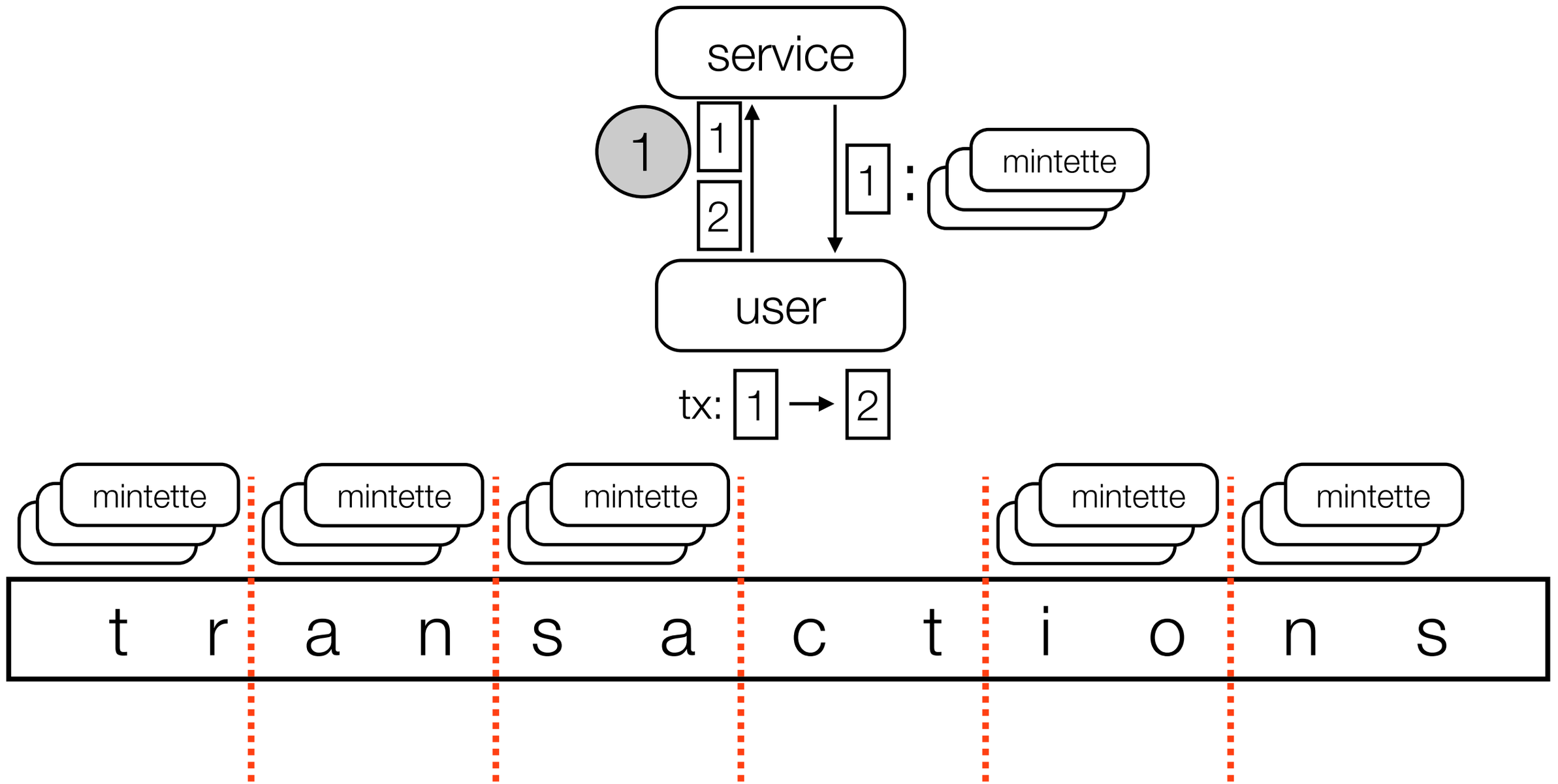

service
1
1
2
user
tx:
1
2

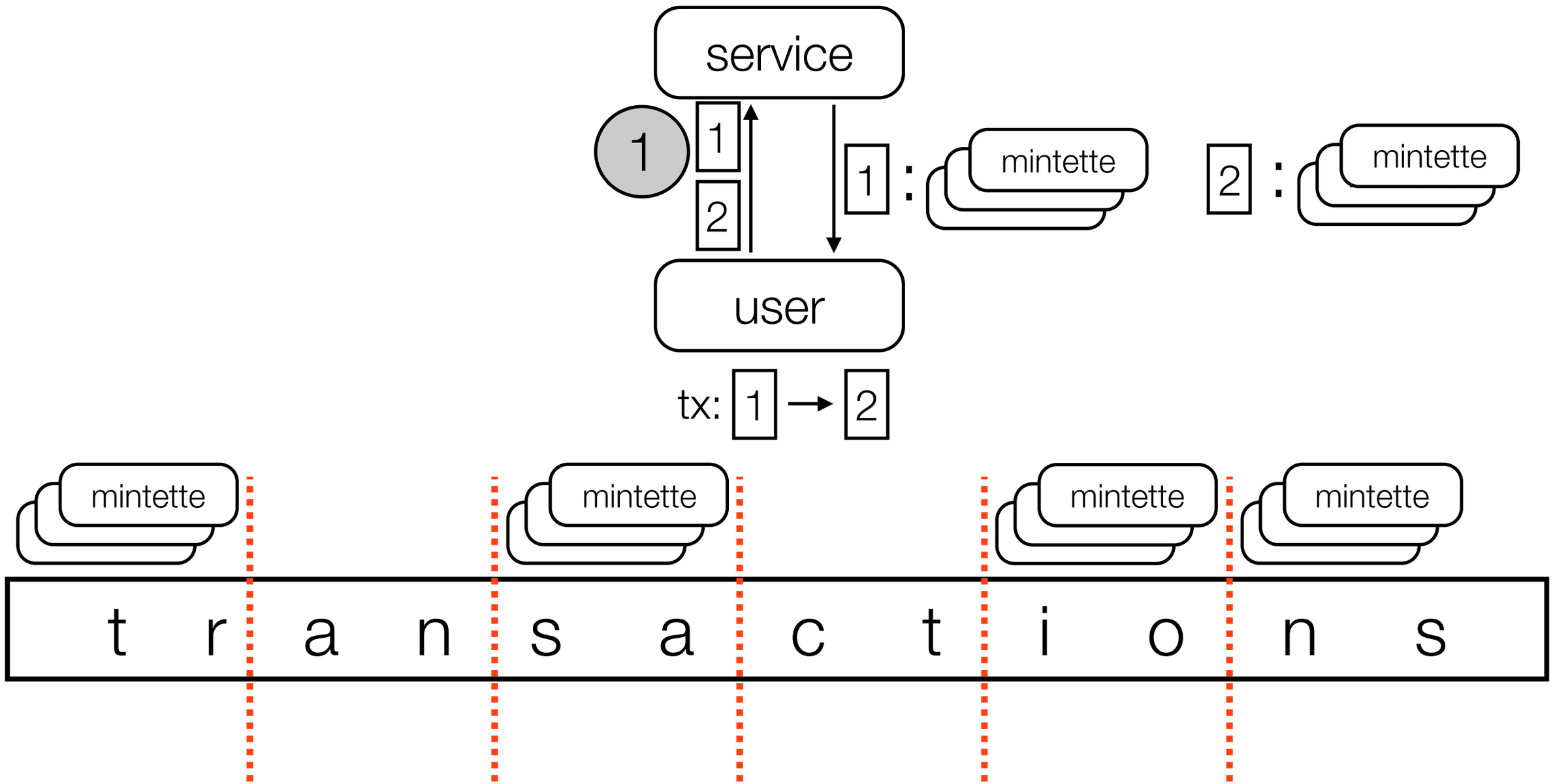


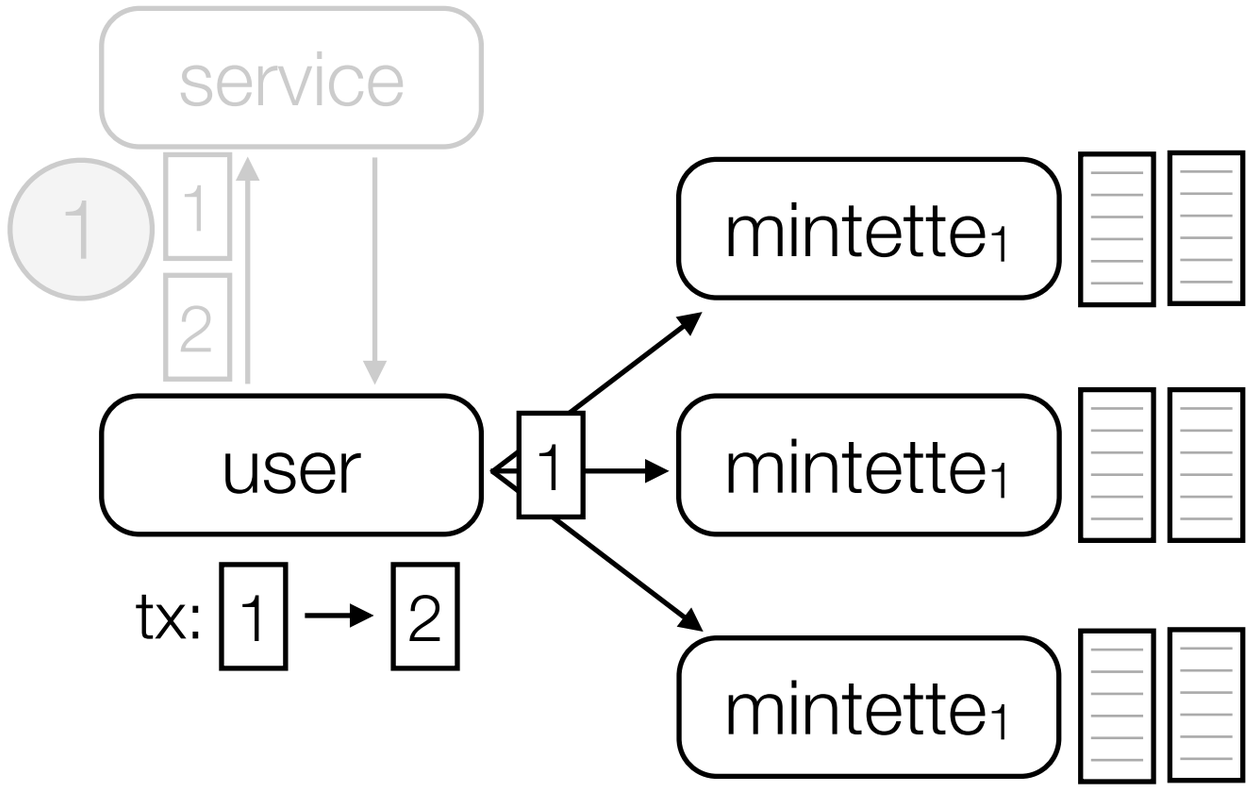
t r a n s a c t i o n s


service
1
1
2
user
tx: 1 → 2
t r a n s a c t i o n s

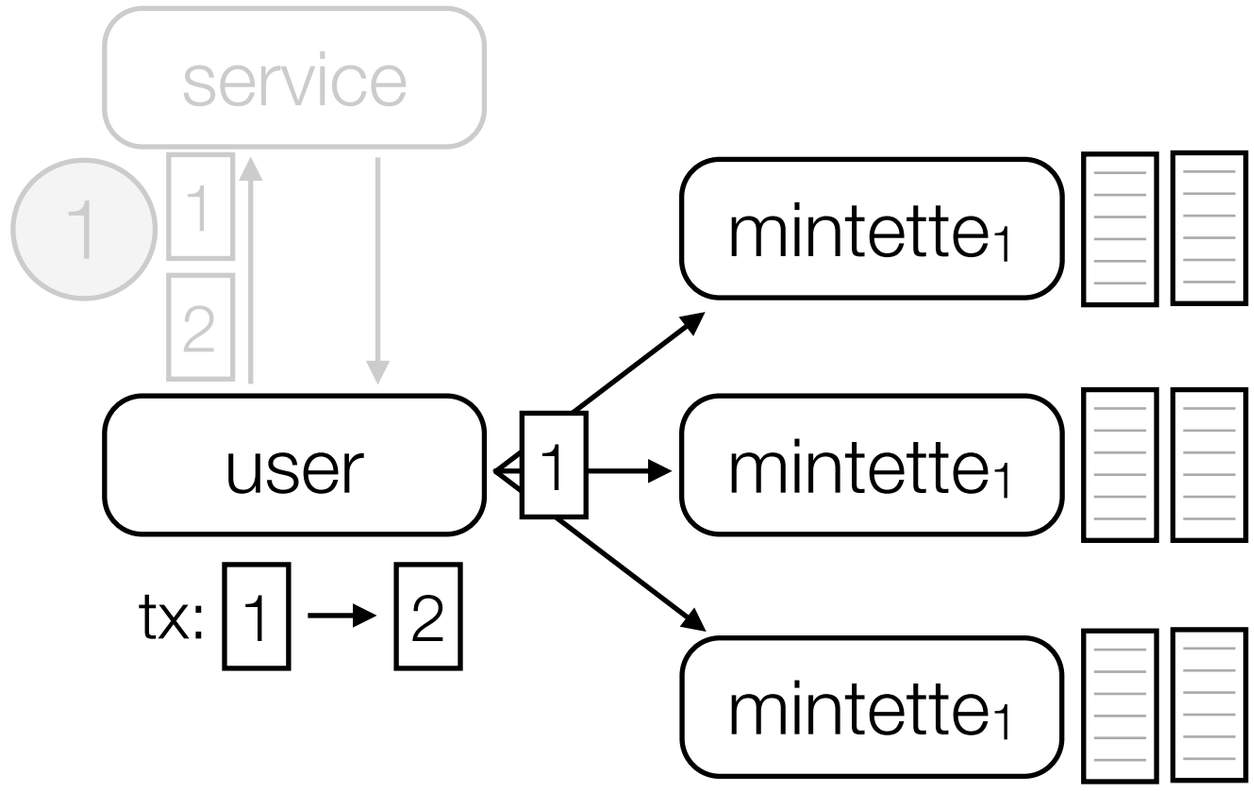

service
1
1
2
user
tx: 1 → 2
mintette
mintette
mintette
mintette
mintette
mintette
t r a n s a c t i o n s


service
1
1
2
1
mintette
user
tx: 1 → 2
mintette
mintette
mintette
mintette
mintette
transactions

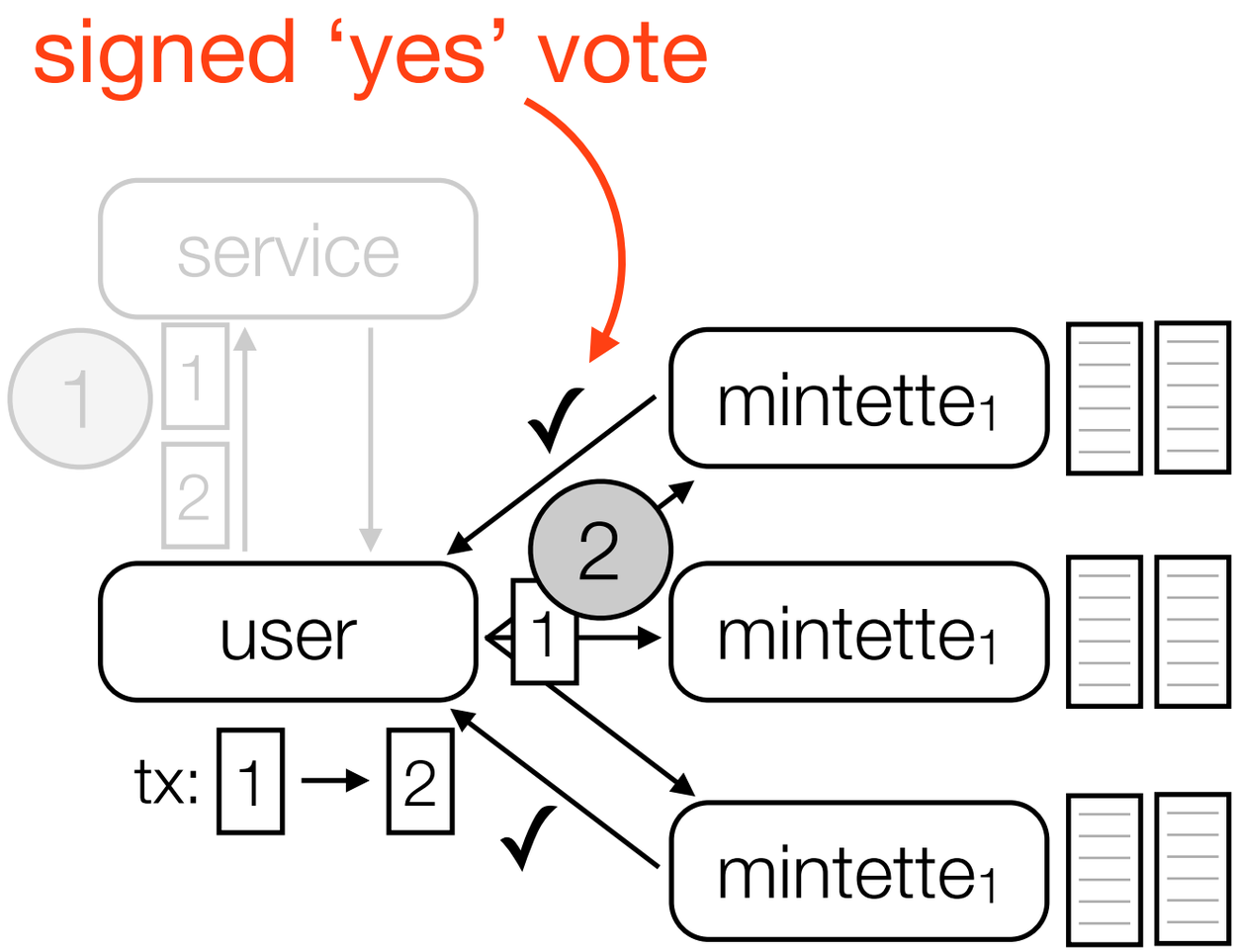

service
1
1
2
1 :
mintette
2 :
mintette
user
tx: 1 → 2
mintette
mintette
mintette
mintette
t r a n s a c t i o n s

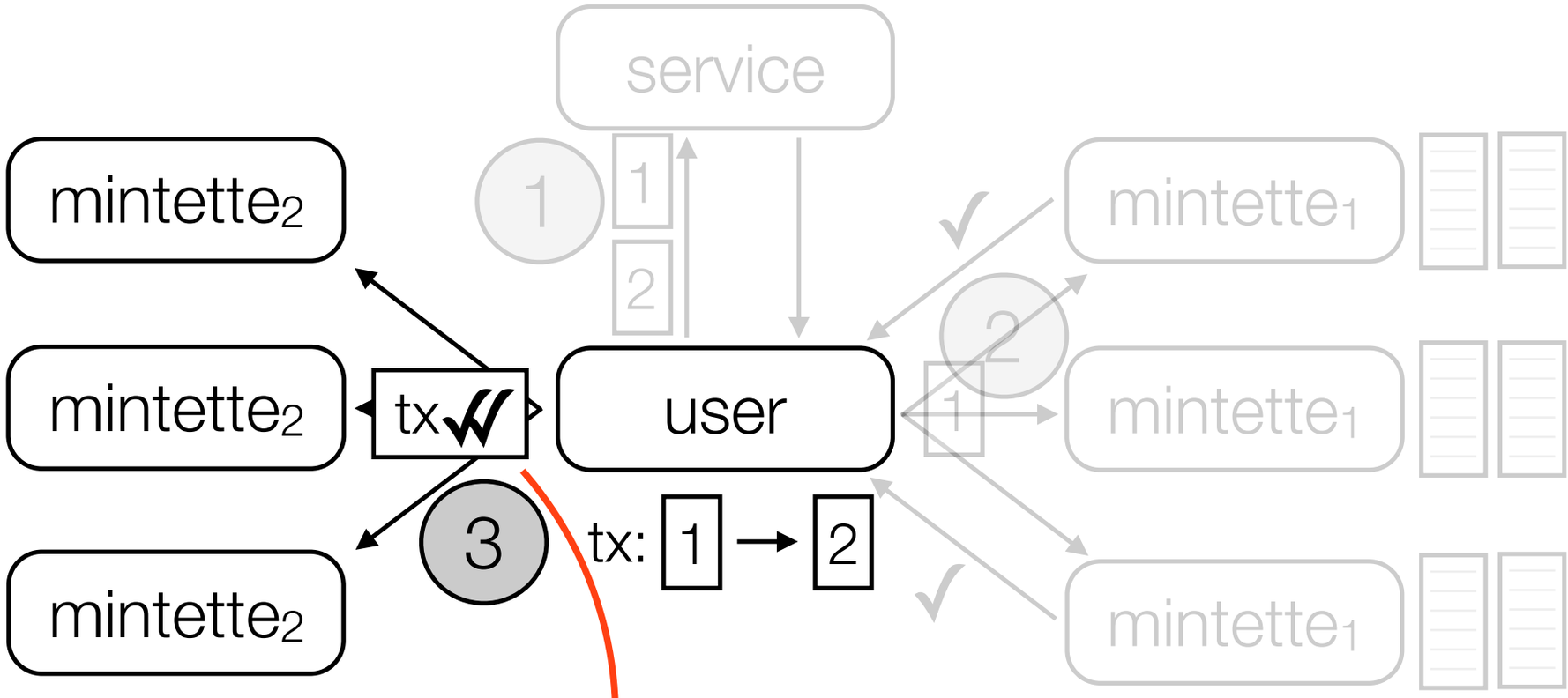

service
1
1
2
user
1
tx:
1
2
mintette1
mintette1
mintette1

mintettes check for **double spending**…



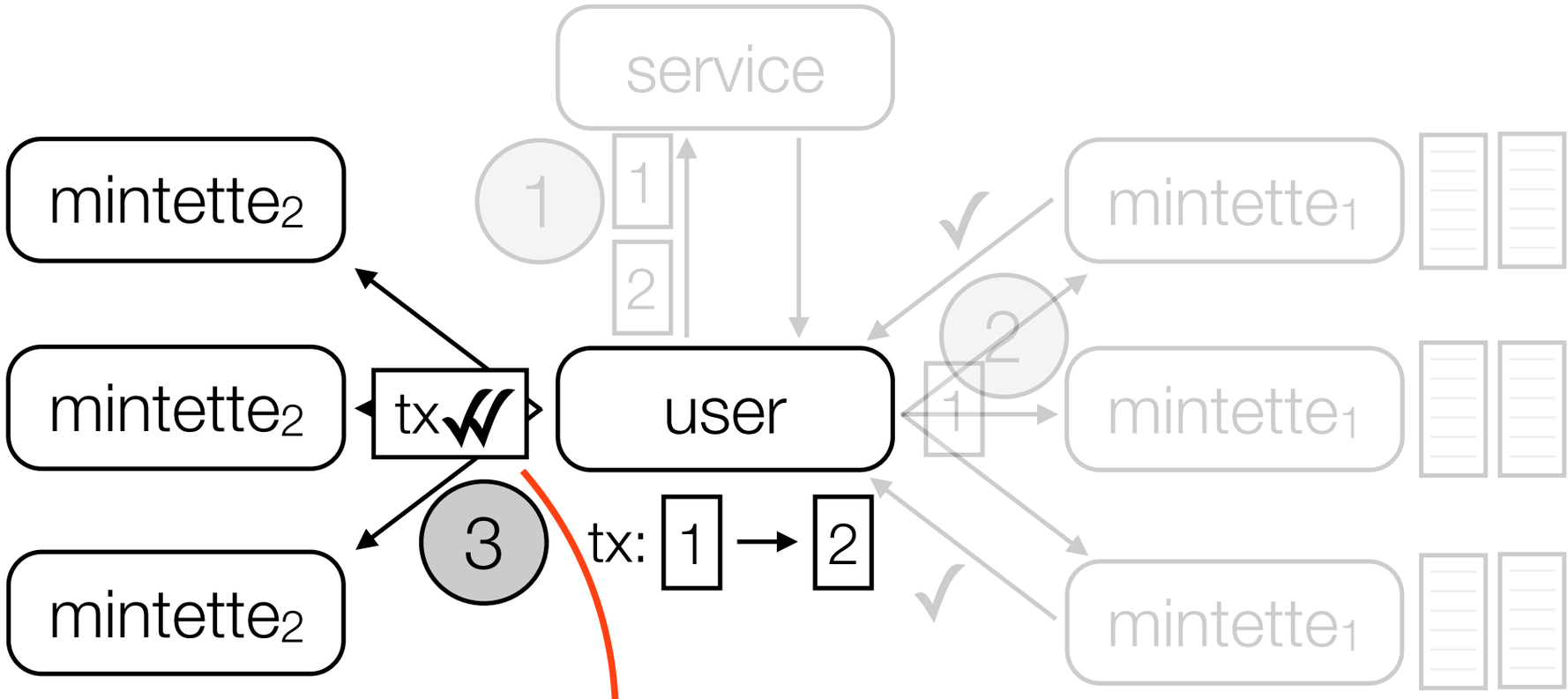
…using lists of **unspent transaction outputs** (utxo)


signed ‘yes’ vote
service
1
1
2
mintette1
2
user
1
mintette1
tx: 1 2
mintette1



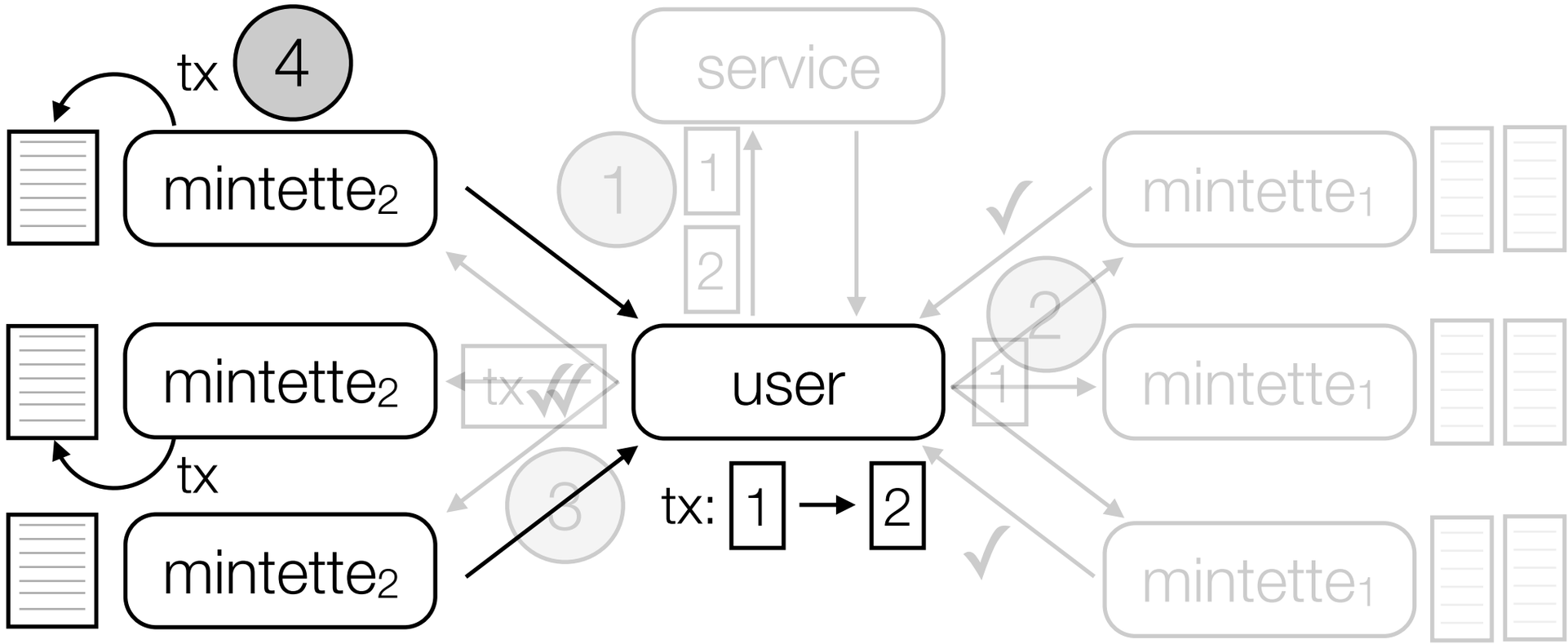
“bundle of evidence” contains ‘yes’ votes
from **majority** of mintettes in shard

mintettes check validity of bundle by checking for signatures from authorized mintettes…



“bundle of evidence” contains ‘yes’ votes from **majority** of mintettes in shard

…and if satisfied they add transaction
to be **committed** and send back **receipt**



security properties

no double spending (if honest majority per shard)

non-repudiation

auditability (if mintettes log their behavior)

consensus features

conceptually simple

no broadcast

mintettes communicate only with users

no expensive hashing!

scalable

consensus features

conceptually simple

no broadcast

mintettes communicate only with users

no expensive hashing!

scalable

↑ computational power ⇒ ↑ throughput

consensus features

T = set of txs generated per second

Q = # mintettes per shard

M = # mintettes

$$\text{comm. per mintette per sec} = \frac{\sum_{tx \in T} 2(m_{tx}+1)Q}{M}$$

consensus features

T = set of txs generated per second

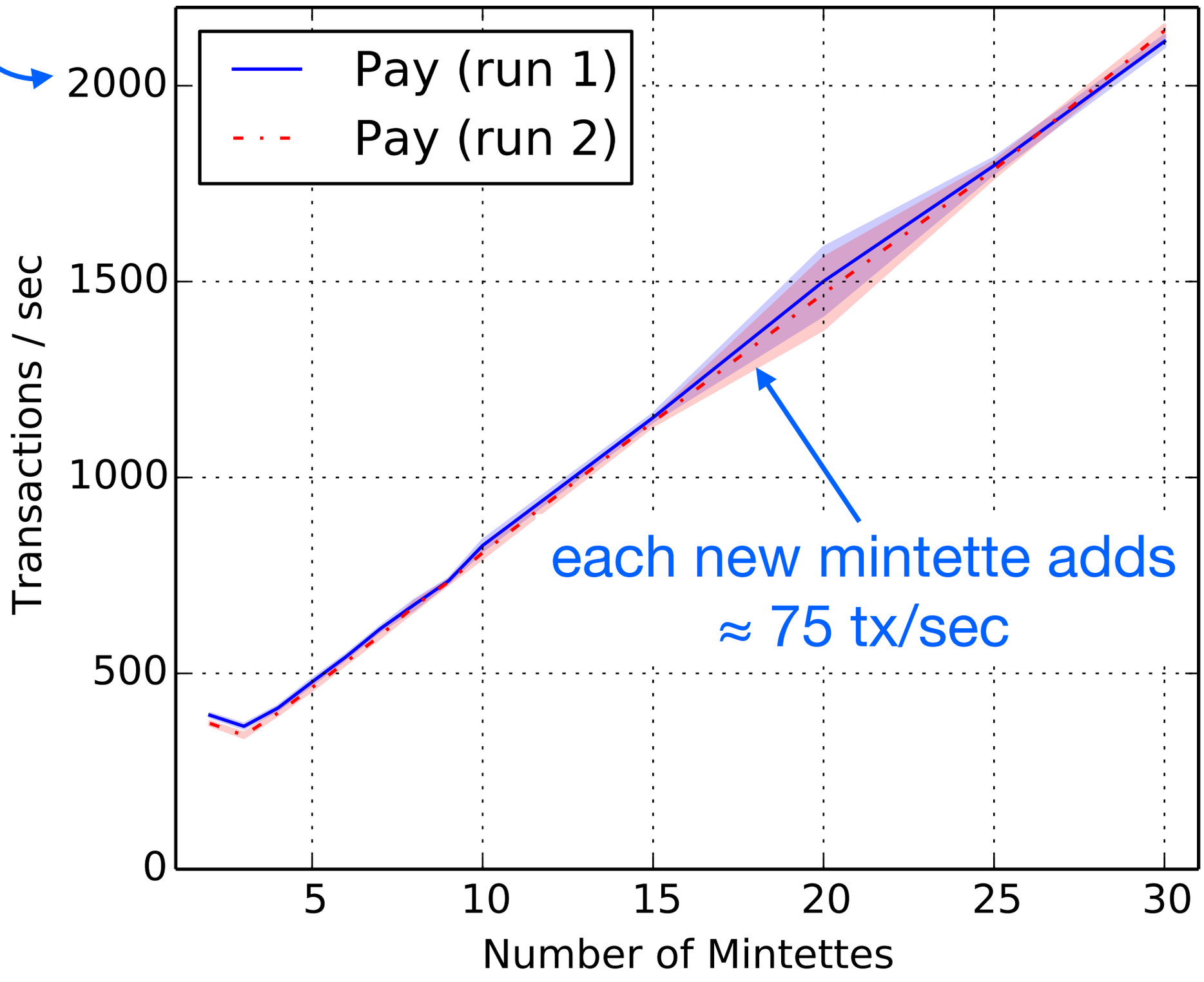
Q = # mintettes per shard

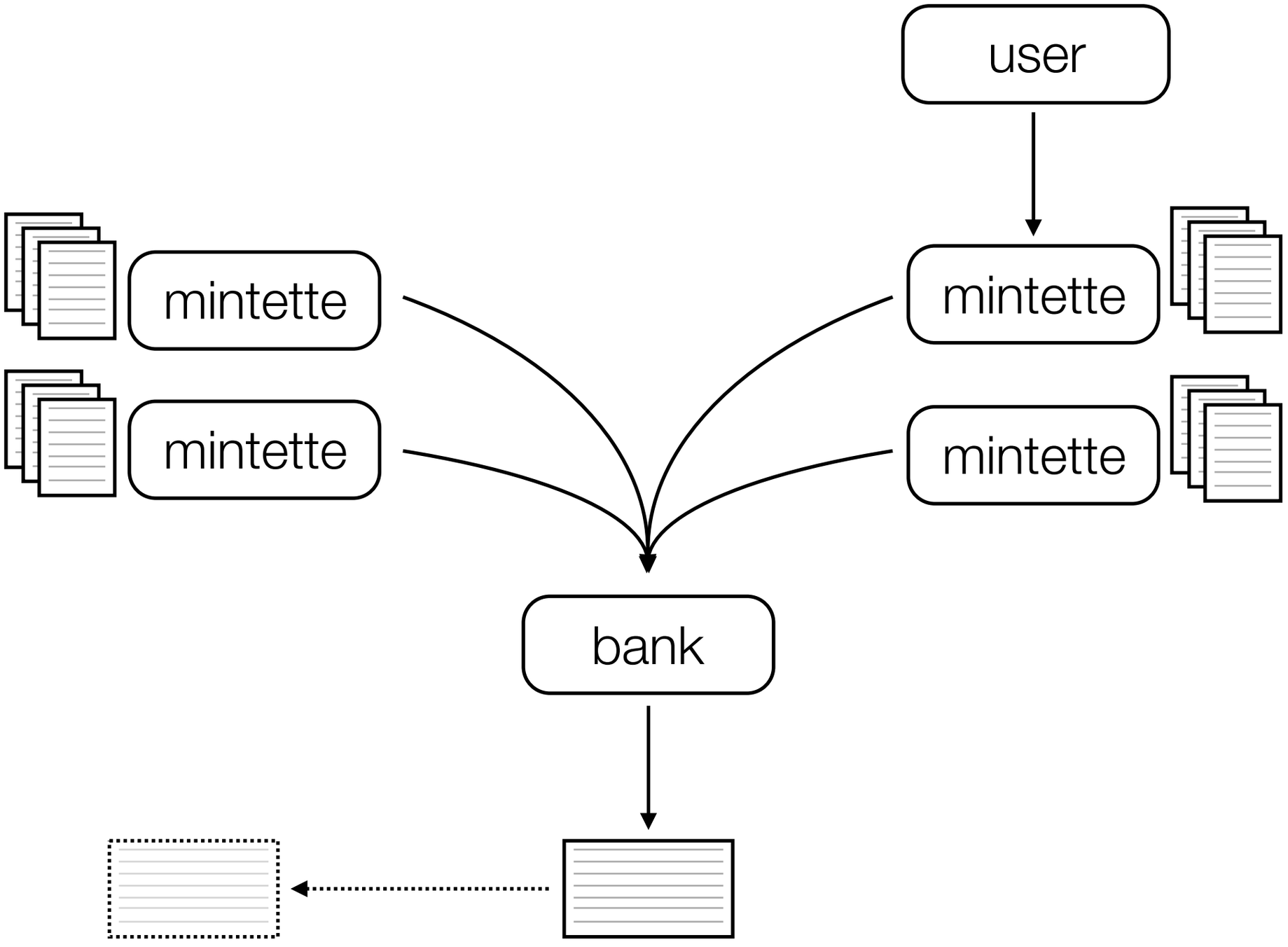
M = # mintettes

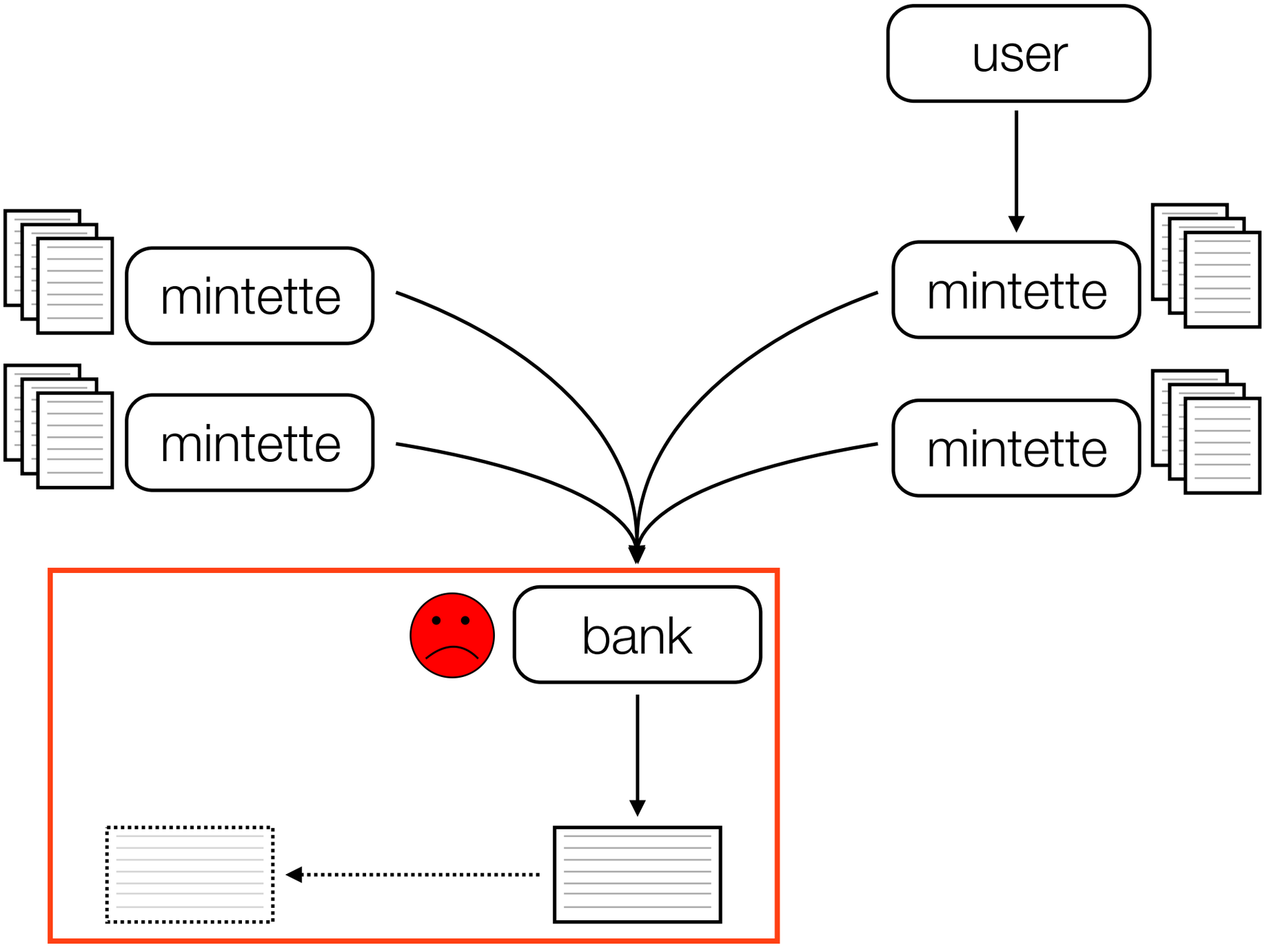
$$\text{comm. per mintette per sec} = \frac{\sum_{tx \in T} 2(m_{tx}+1)Q}{M}$$

scales infinitely as more mintettes are added!

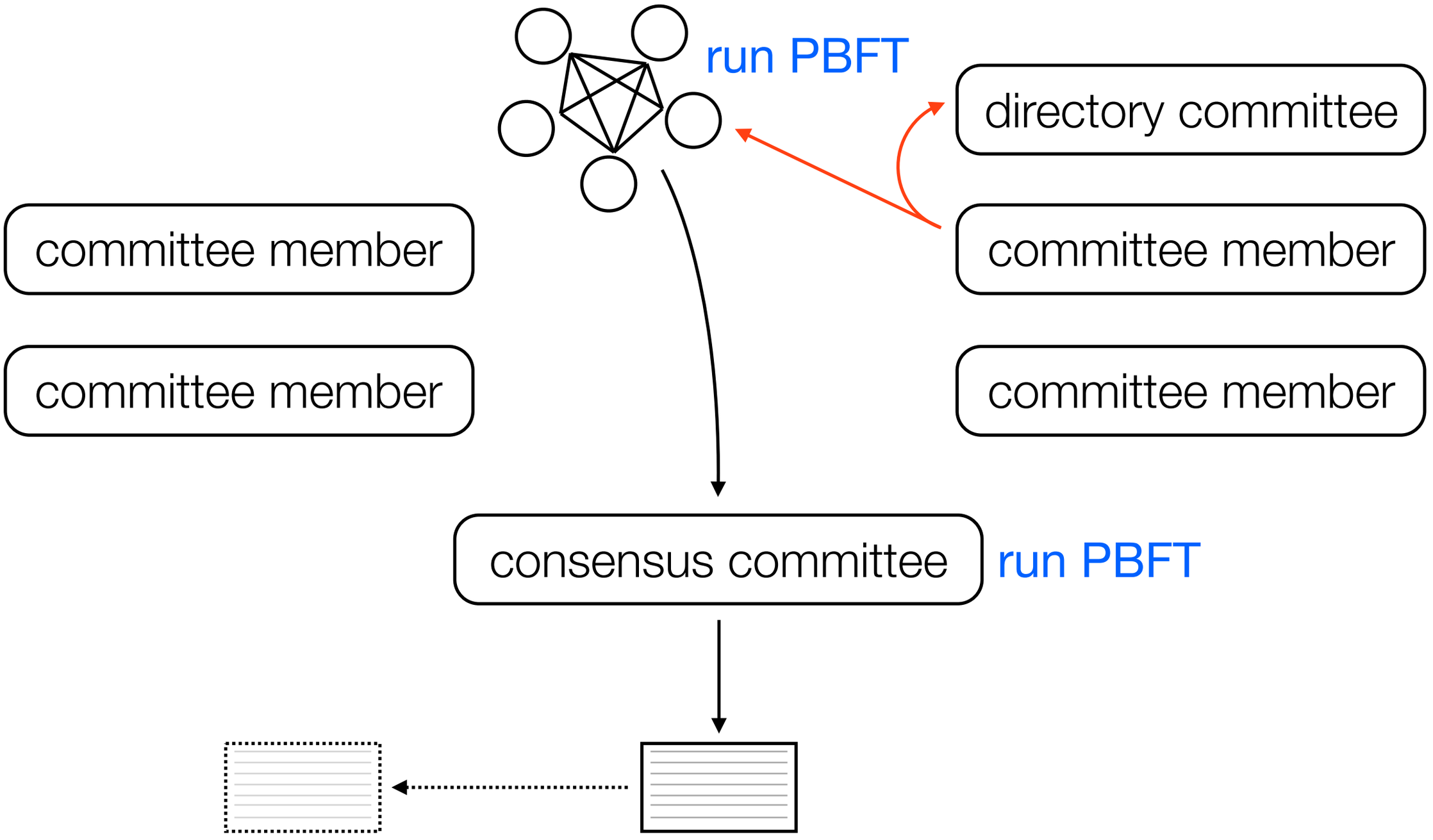
compared to Bitcoin's 7



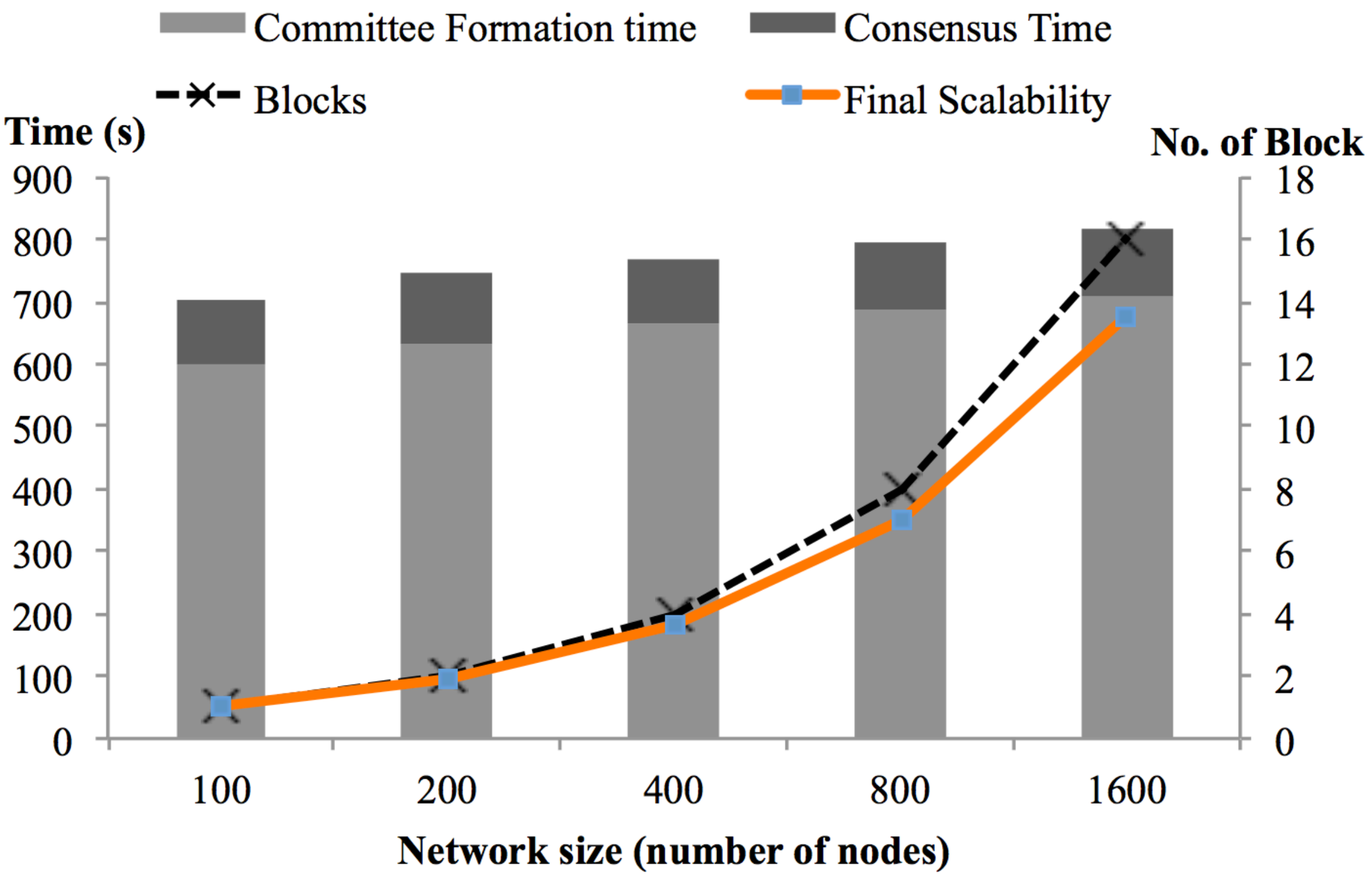

user
mintette
mintette
mintette
mintette
bank


user
mintette
mintette
mintette
mintette
bank

Elastico [LNZBGS CCS'16]



Elastico [LNZBGS CCS'16]



10 usability

9 governance

8 comparisons

7 key management

6 agility

5 interoperability

4 scalability

3 cost-effectiveness

2 privacy

1 scalability

10 usability

9 governance

8 comparisons

7 key management

6 agility

5 interoperability

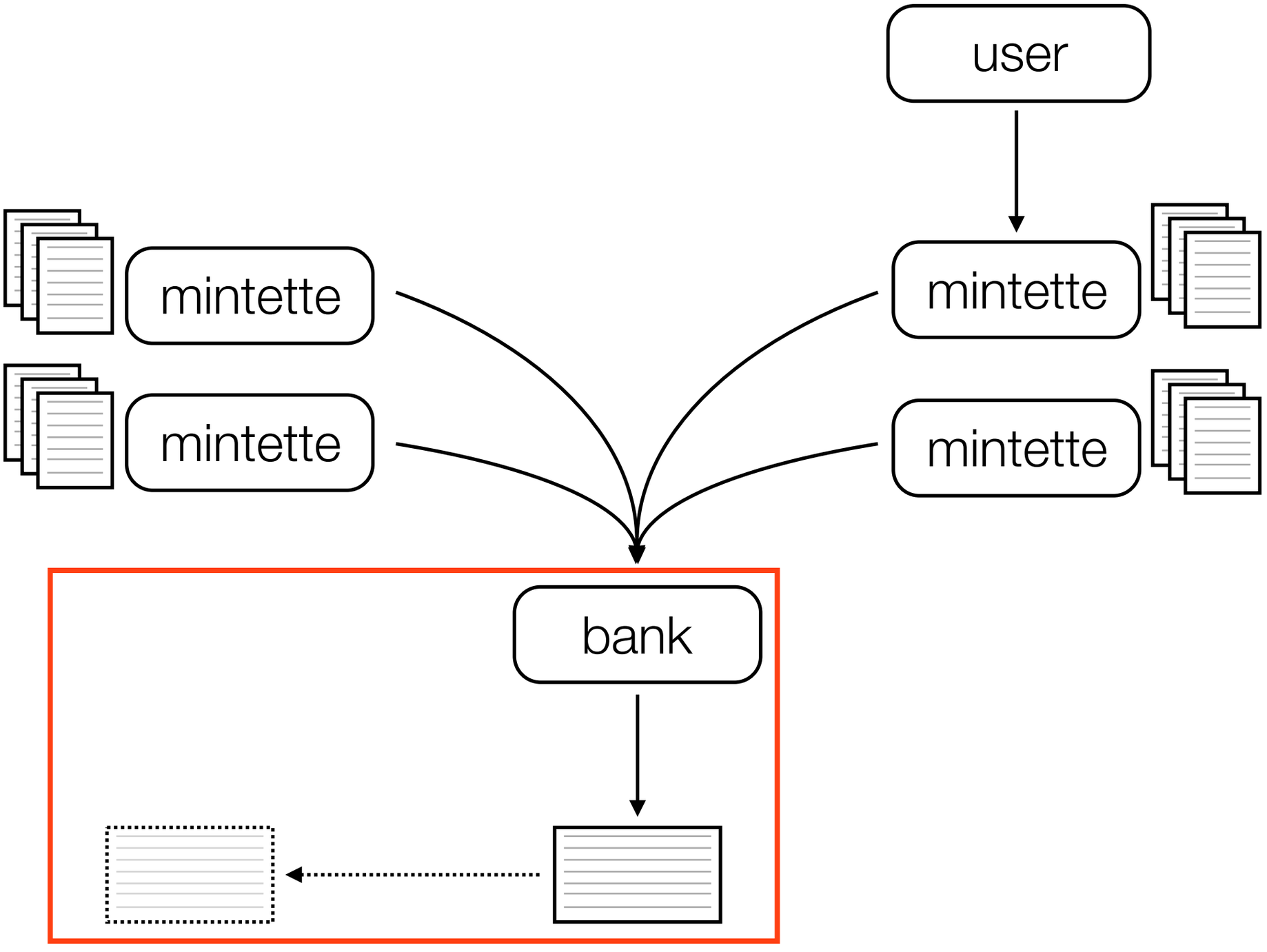
4 scalability

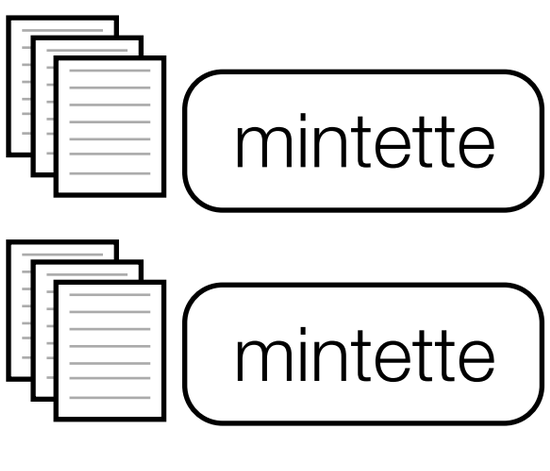
3 cost-effectiveness

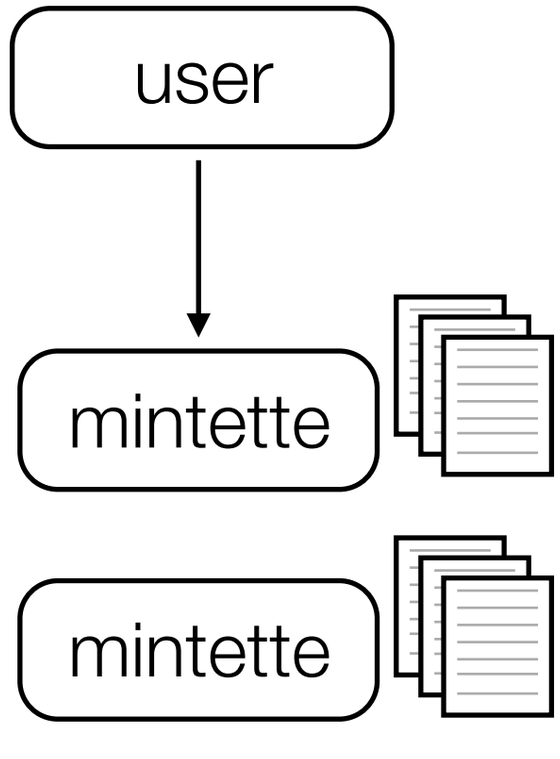
2 privacy

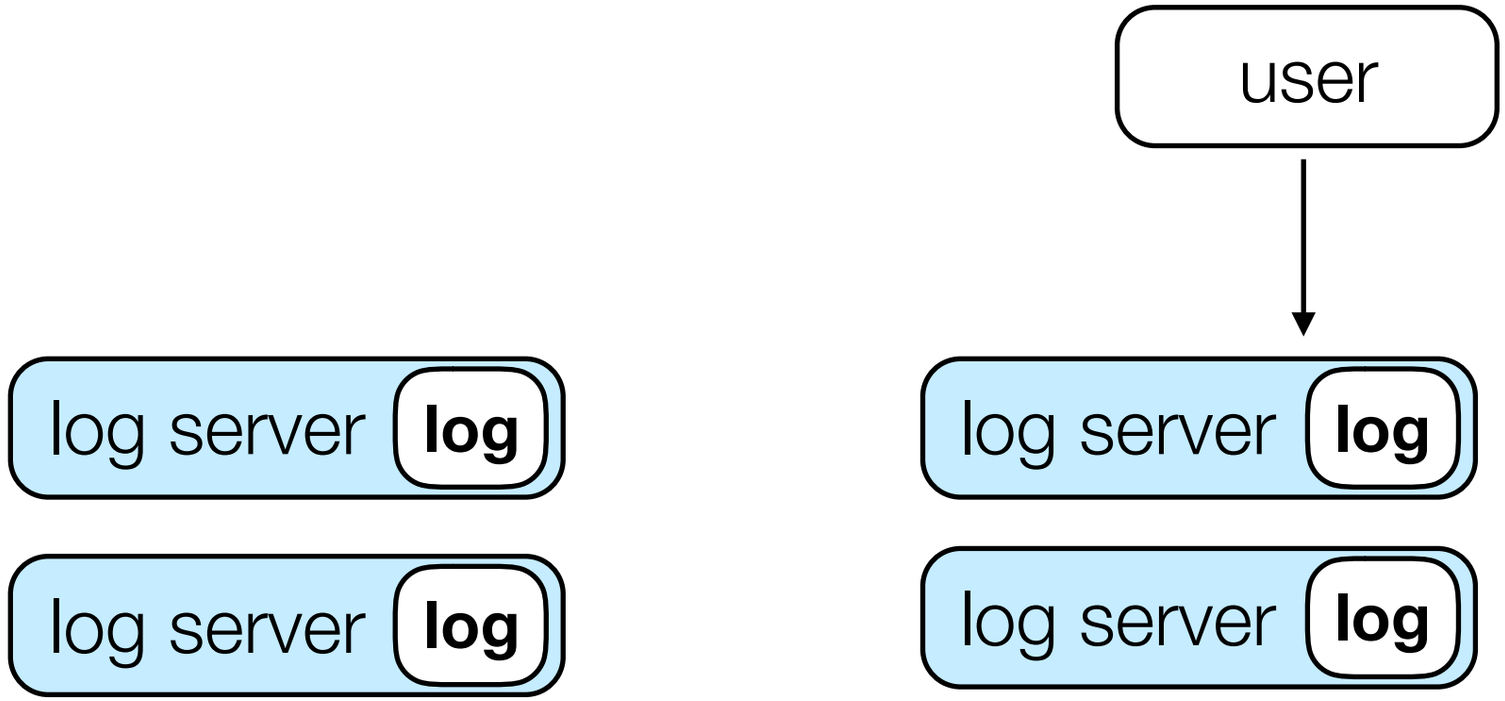
1 scalability

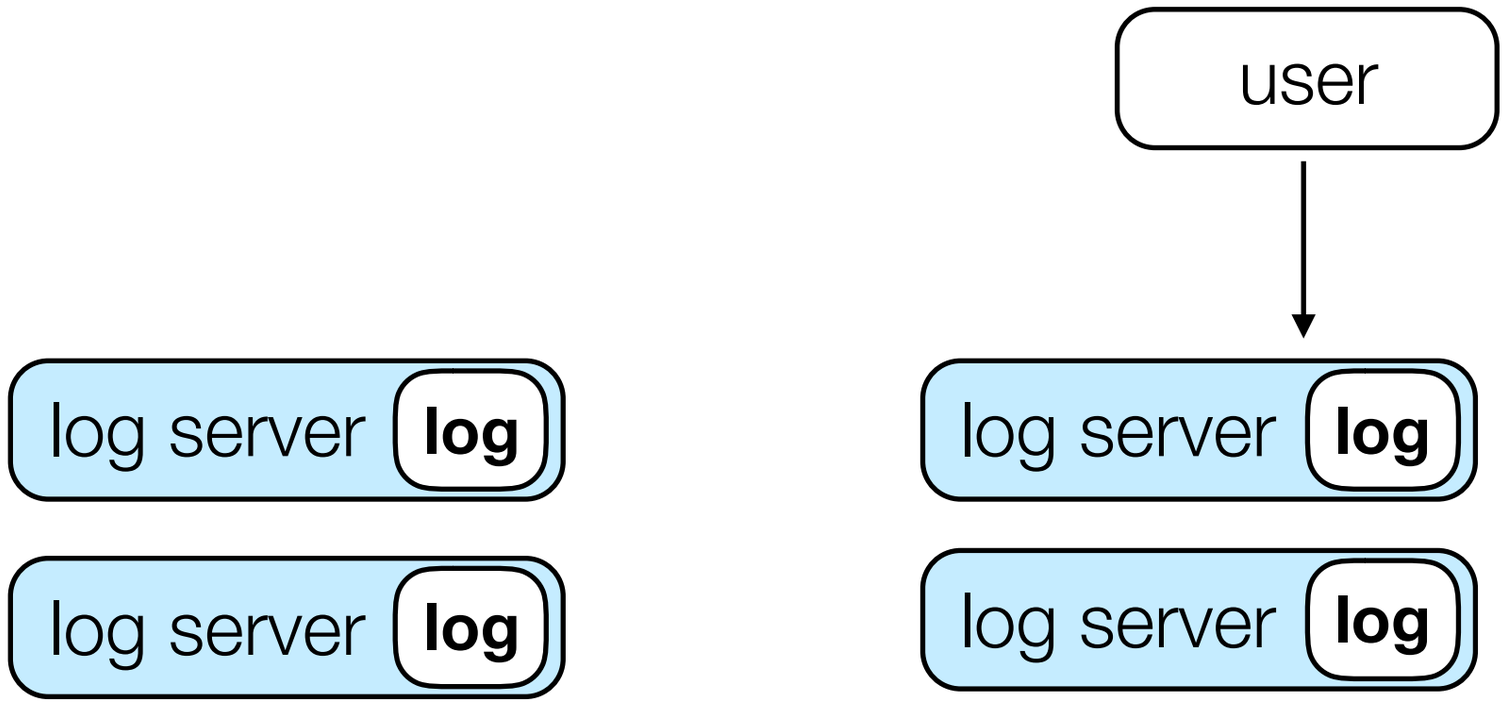
RSCoin [D**M** NDSS'16]



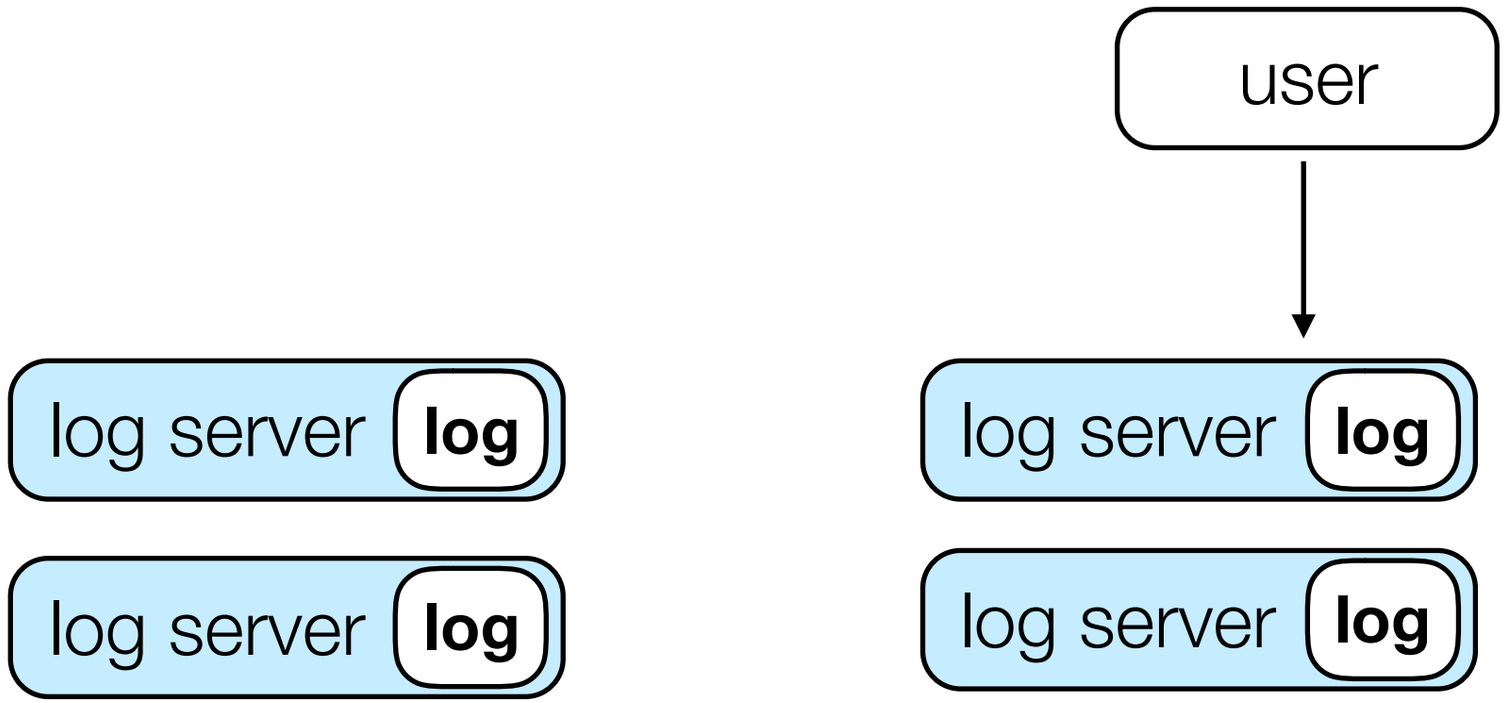

mintette
mintette


user
mintette
mintette


user
log server
log
log server
log
log server
log
log server
log



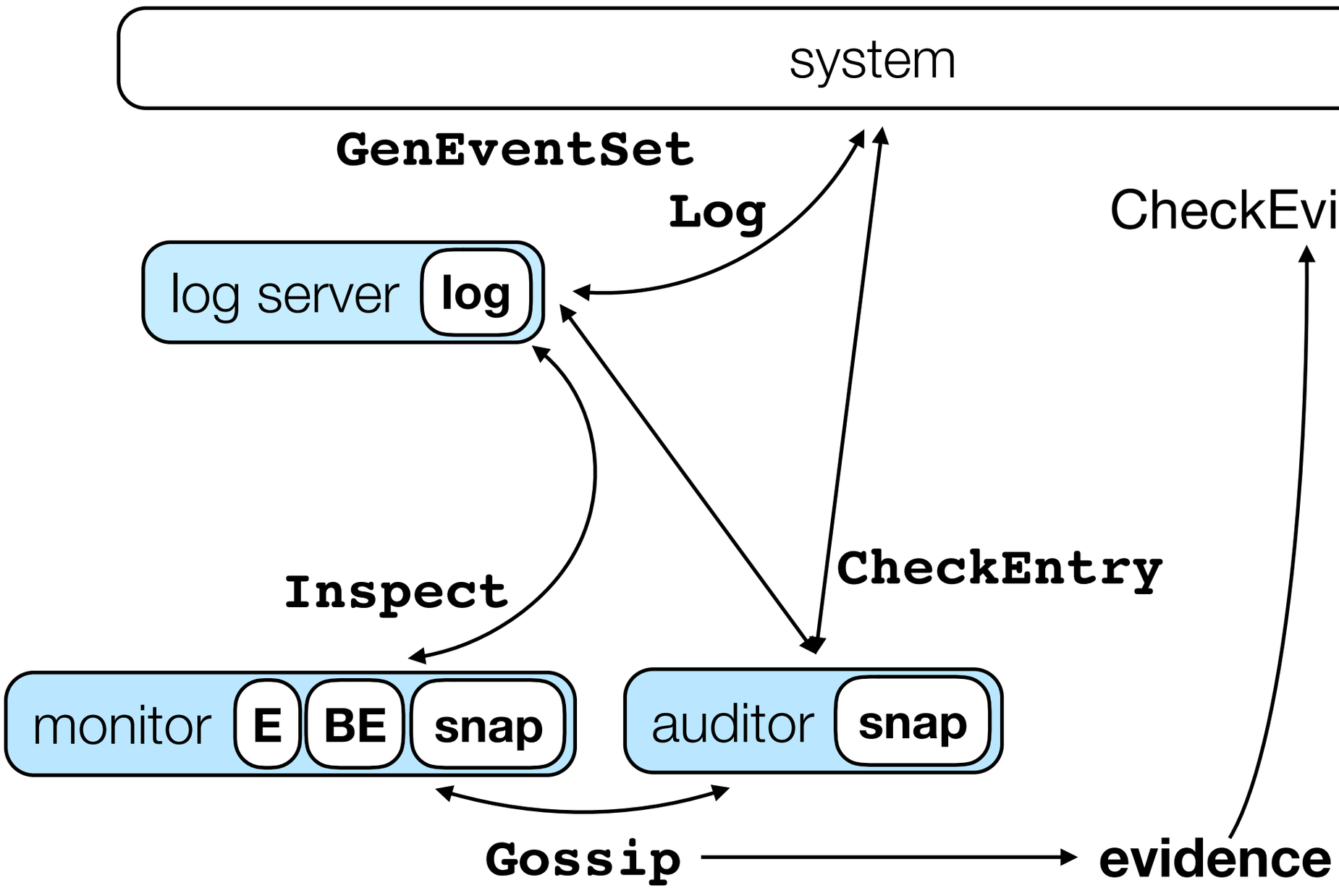
no unified log ⇒ no need for consensus

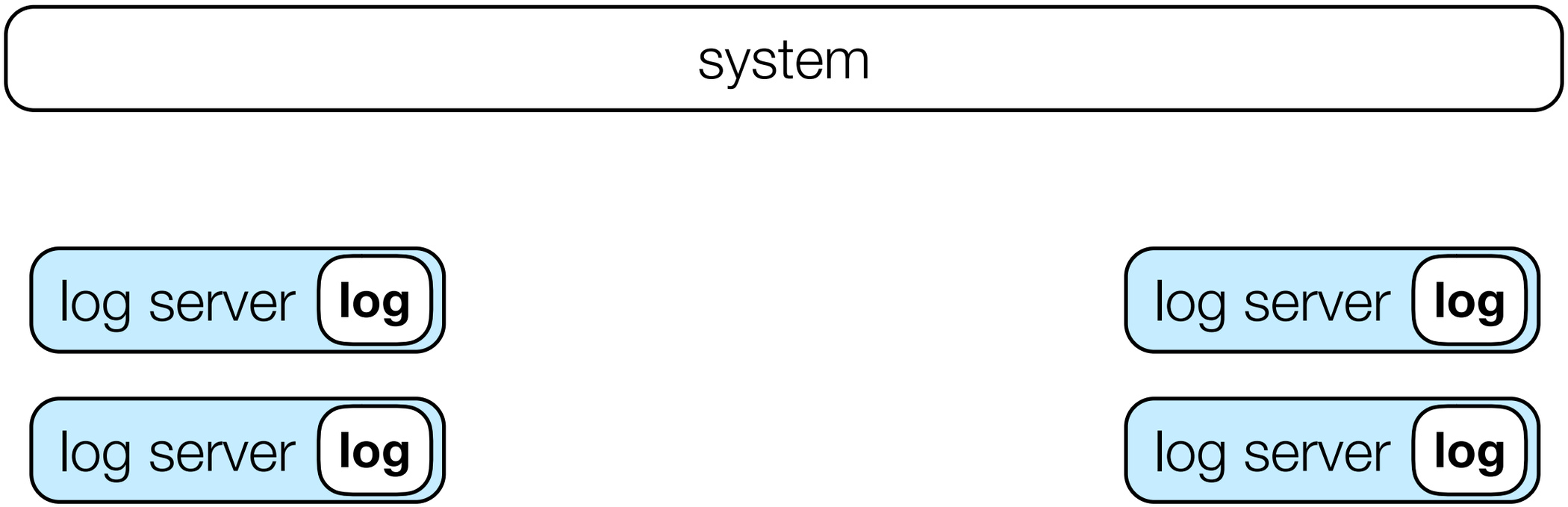


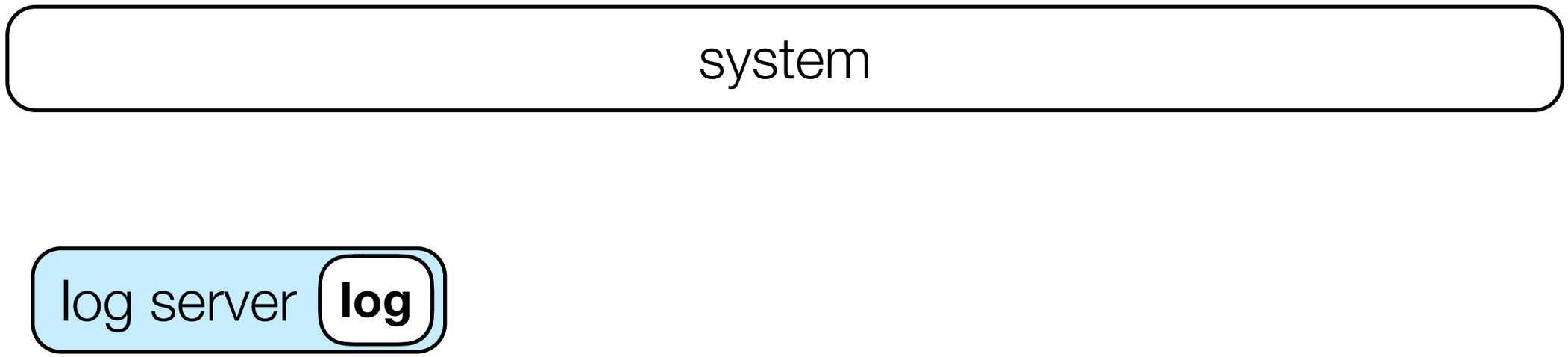
no unified log ⇒ no need for consensus

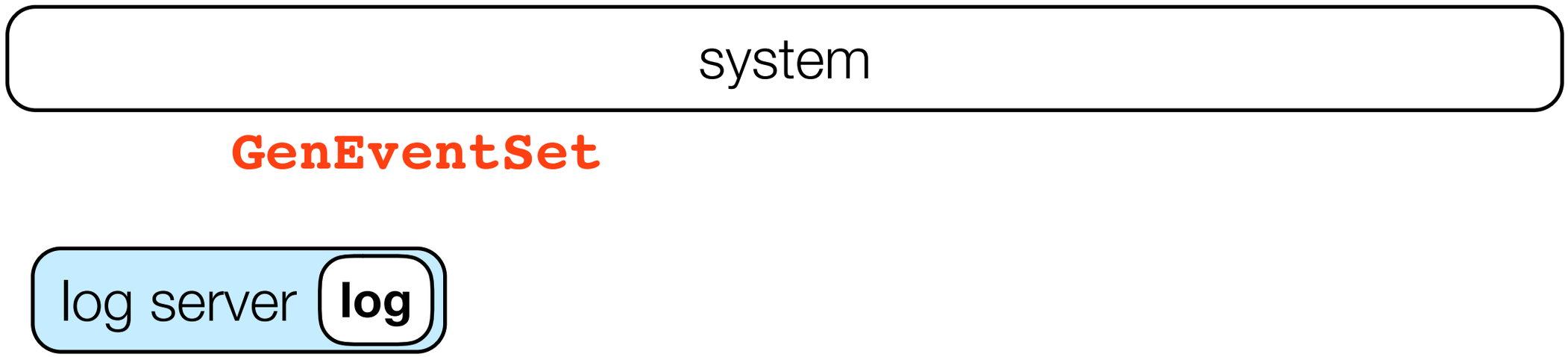
can (retroactively) detect inconsistencies between logs

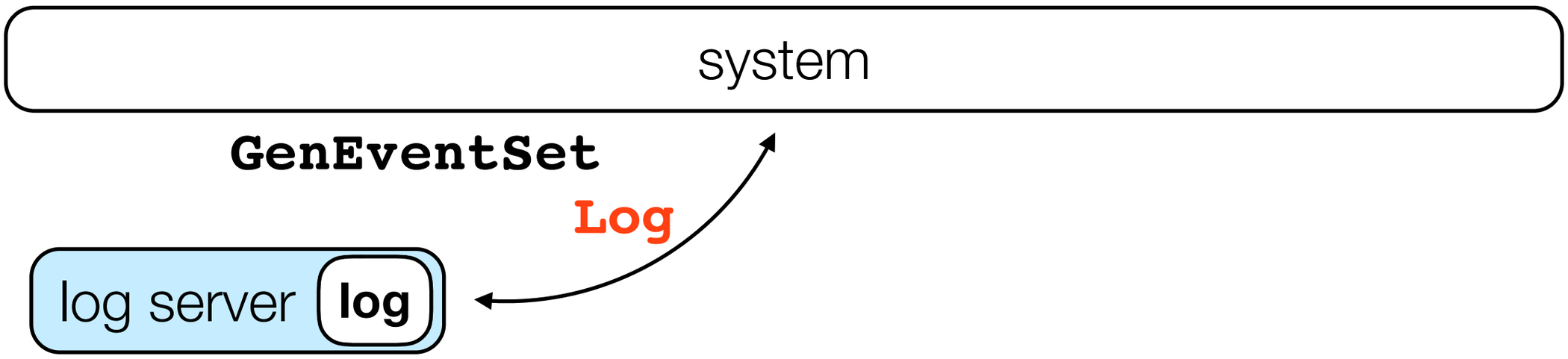
transparency overlays [C**M** CCS'16]

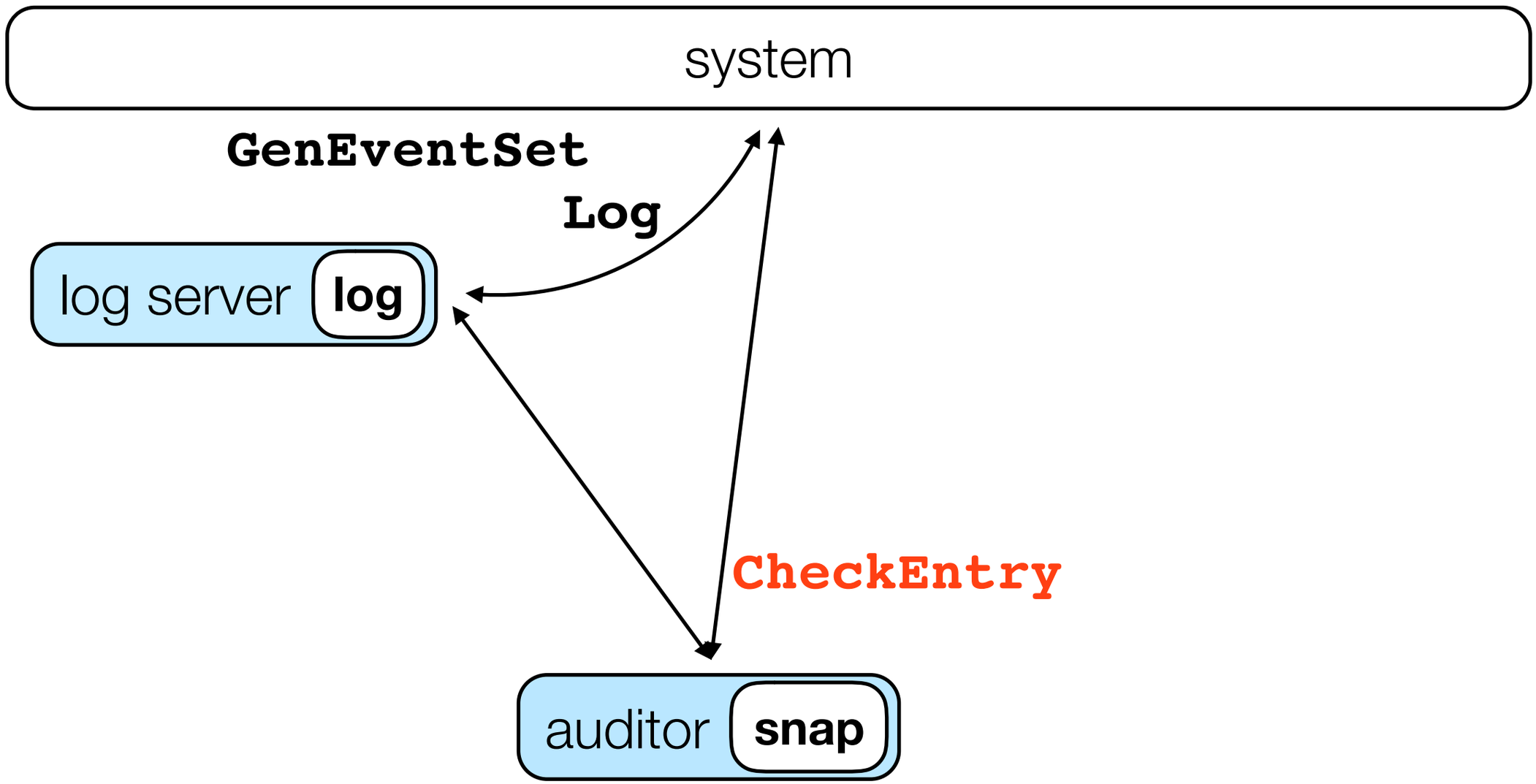



system
log server
log
log server
log
log server
log
log server
log


system
log server
log

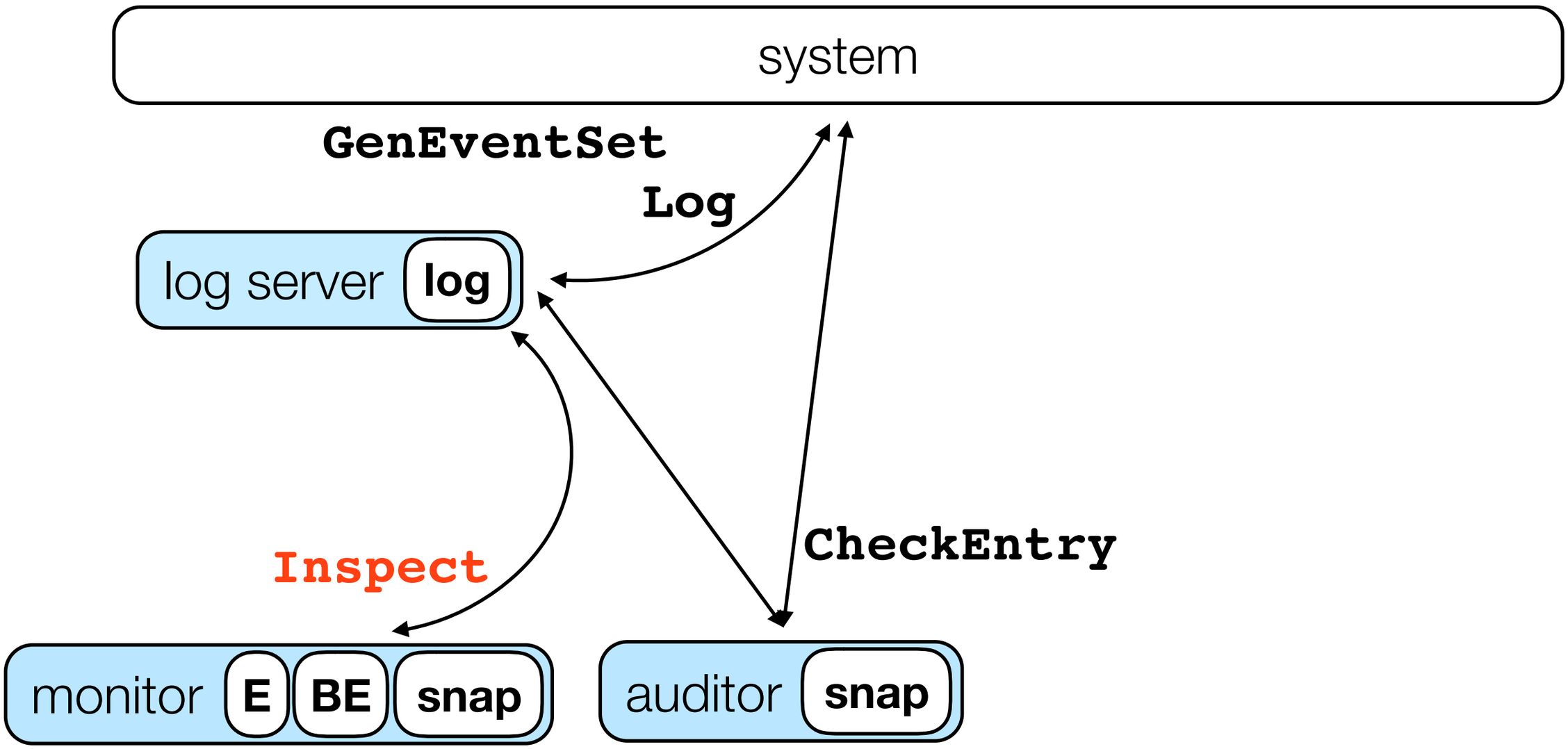

system
GenEventSet
log server
log


system
GenEventSet
Log
log server
log



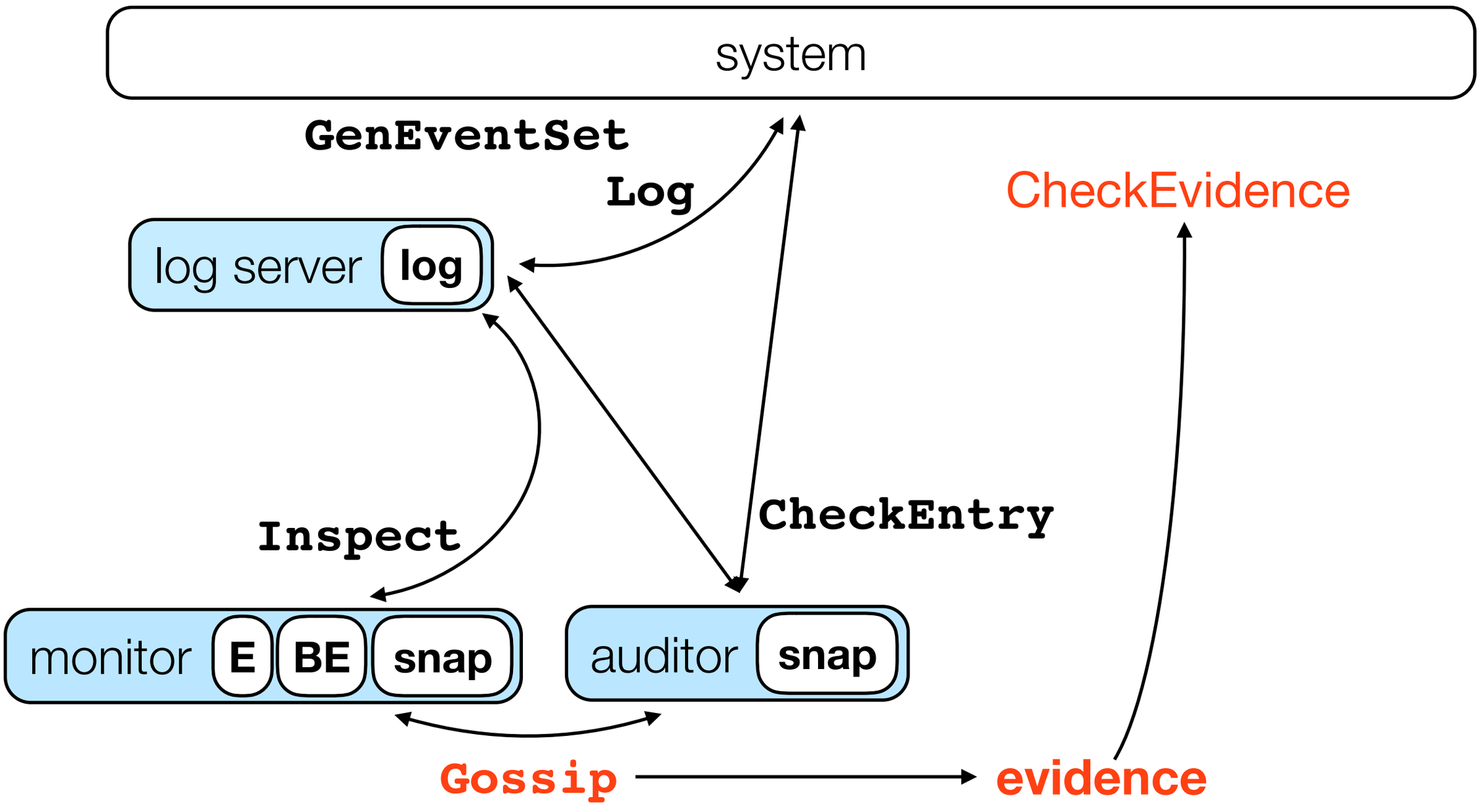
(meaning |snap| ≪ |log|)

auditors (efficiently) determine if events are in the log



(meaning |E| ≈ |log|)

monitors (inefficiently) detect bad events in the log



auditors and monitors ensure consistent view of log

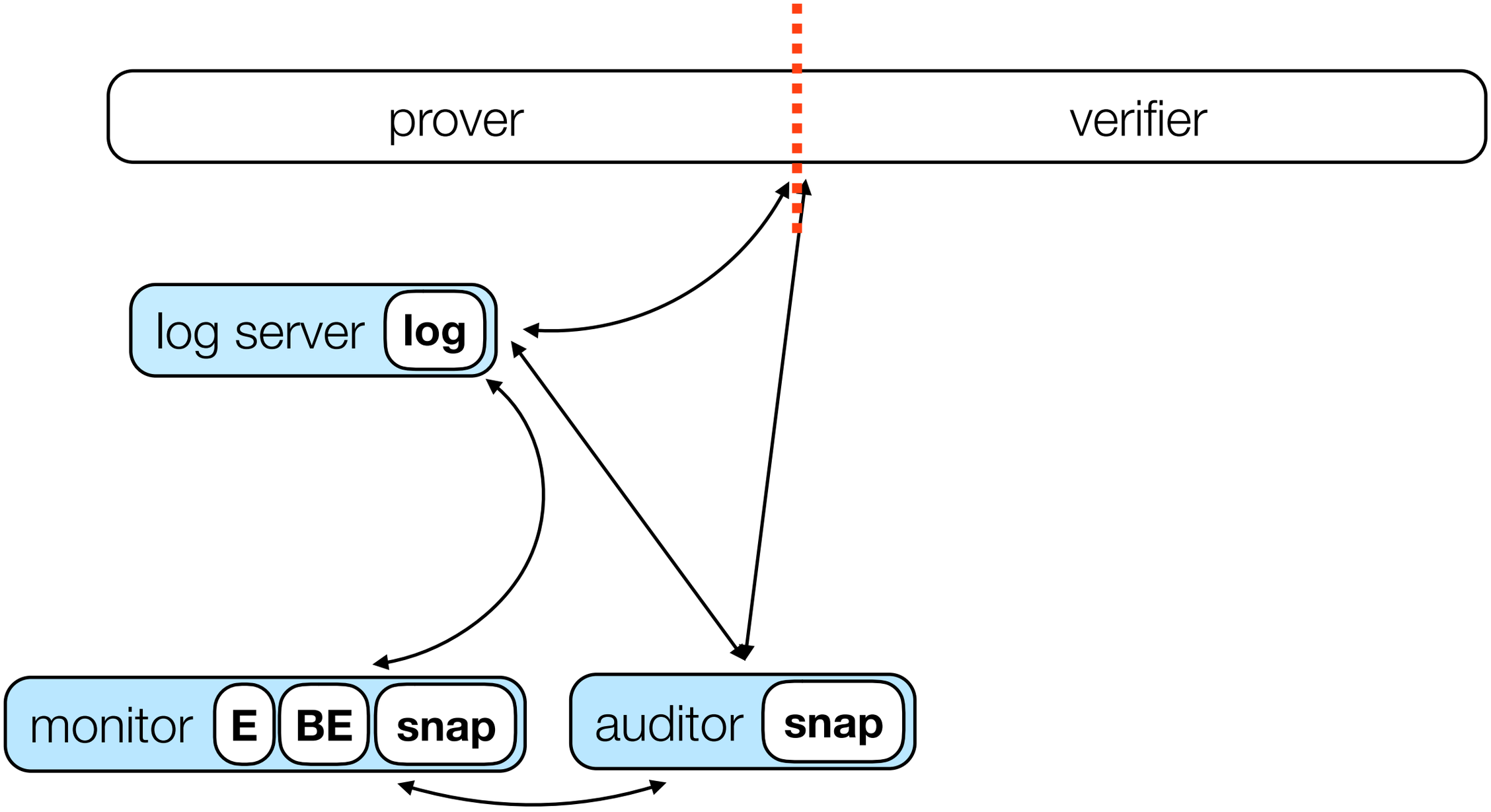
(can output evidence of inconsistencies)

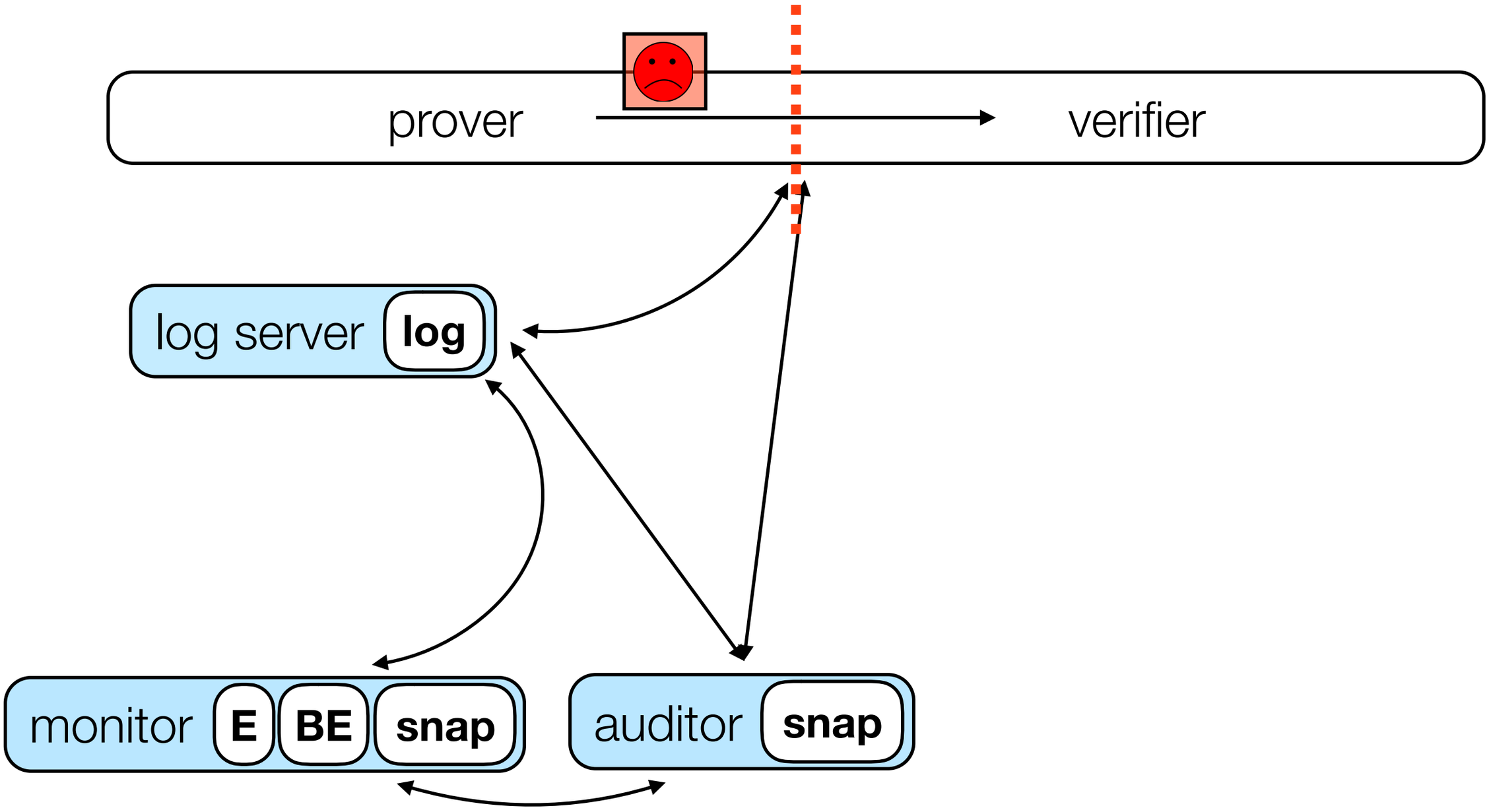
security properties

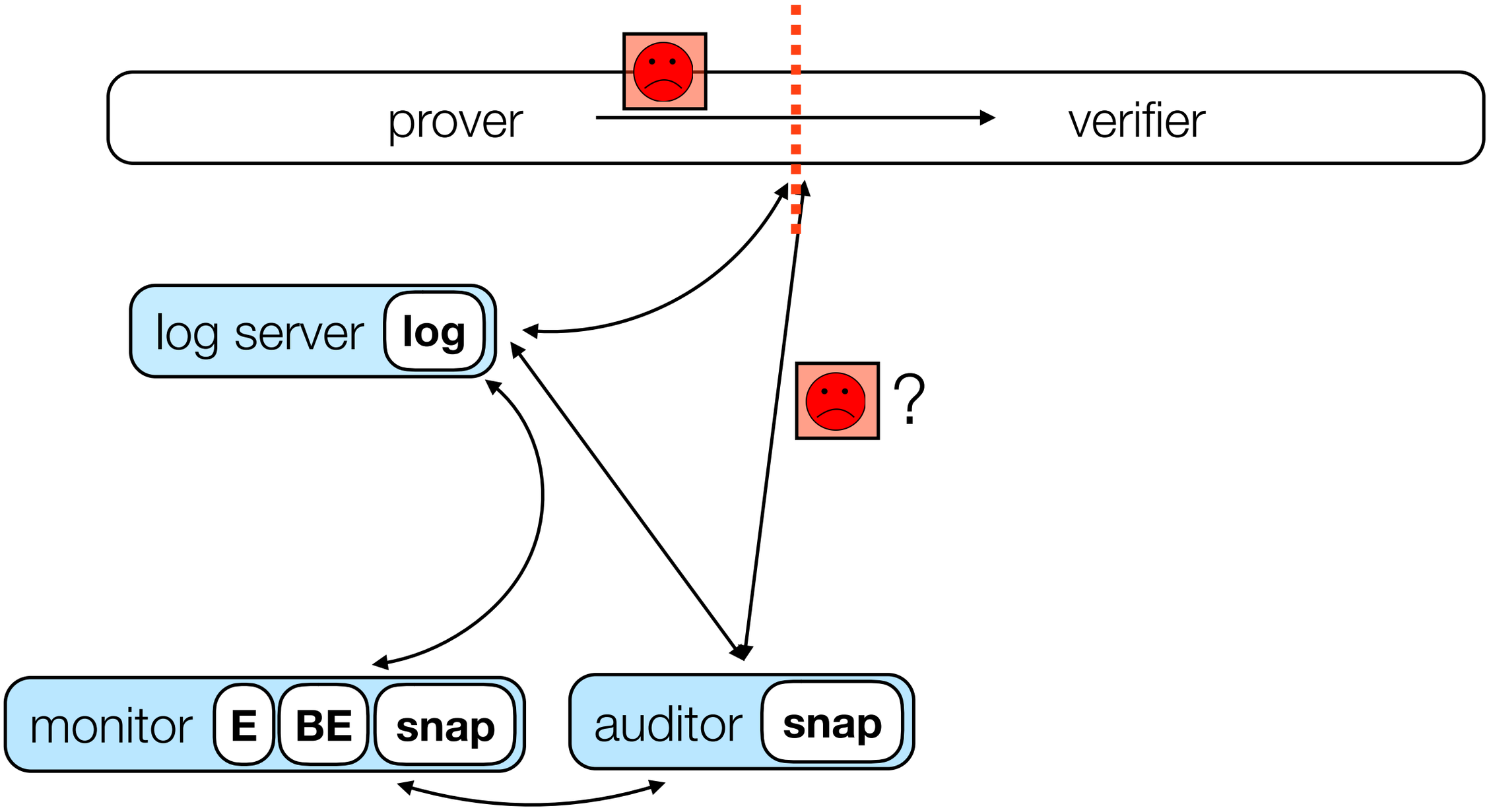
consistency: log server can’t offer different views of log

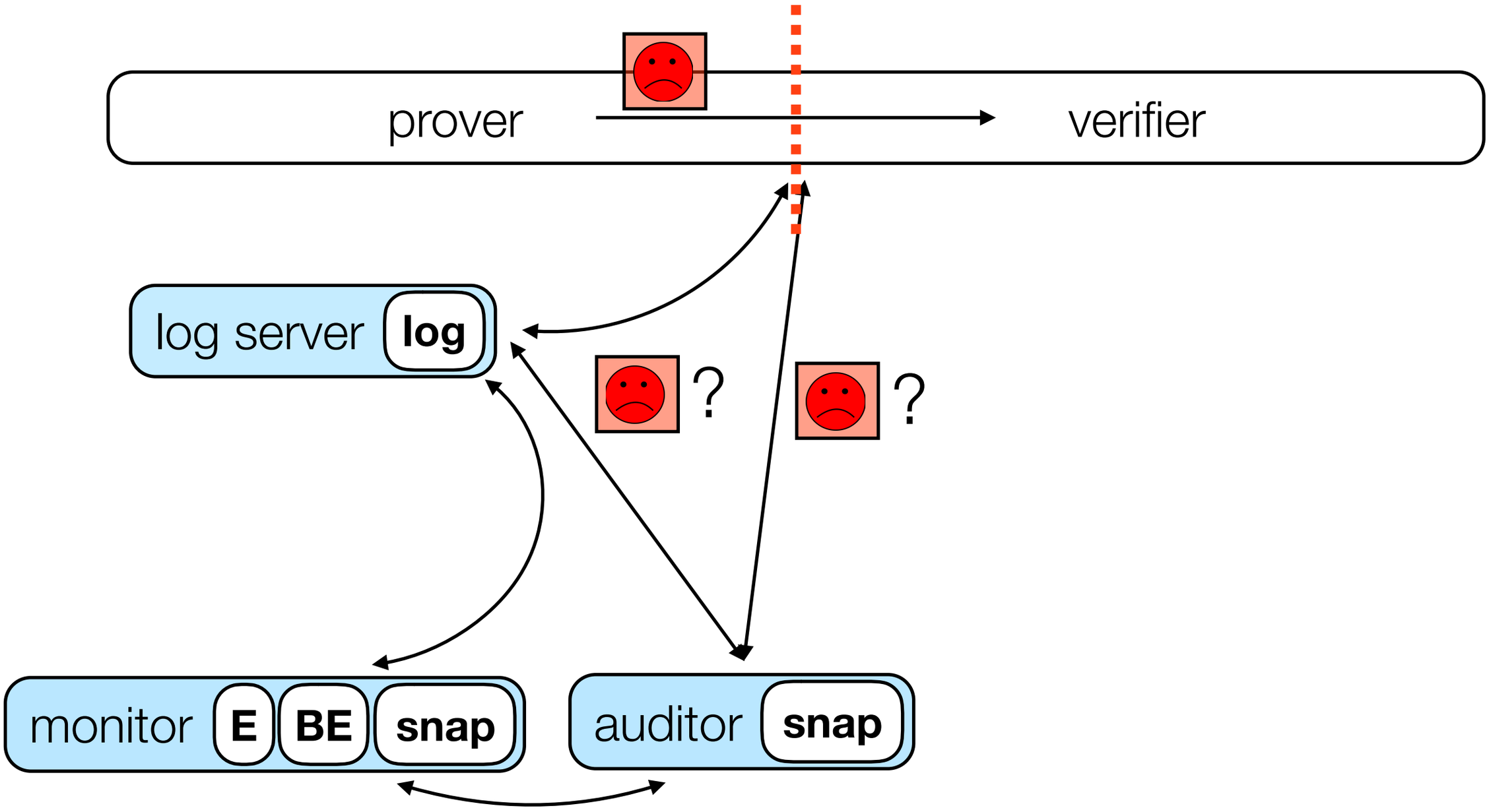
non-frameability: auditor and monitor can’t frame the log

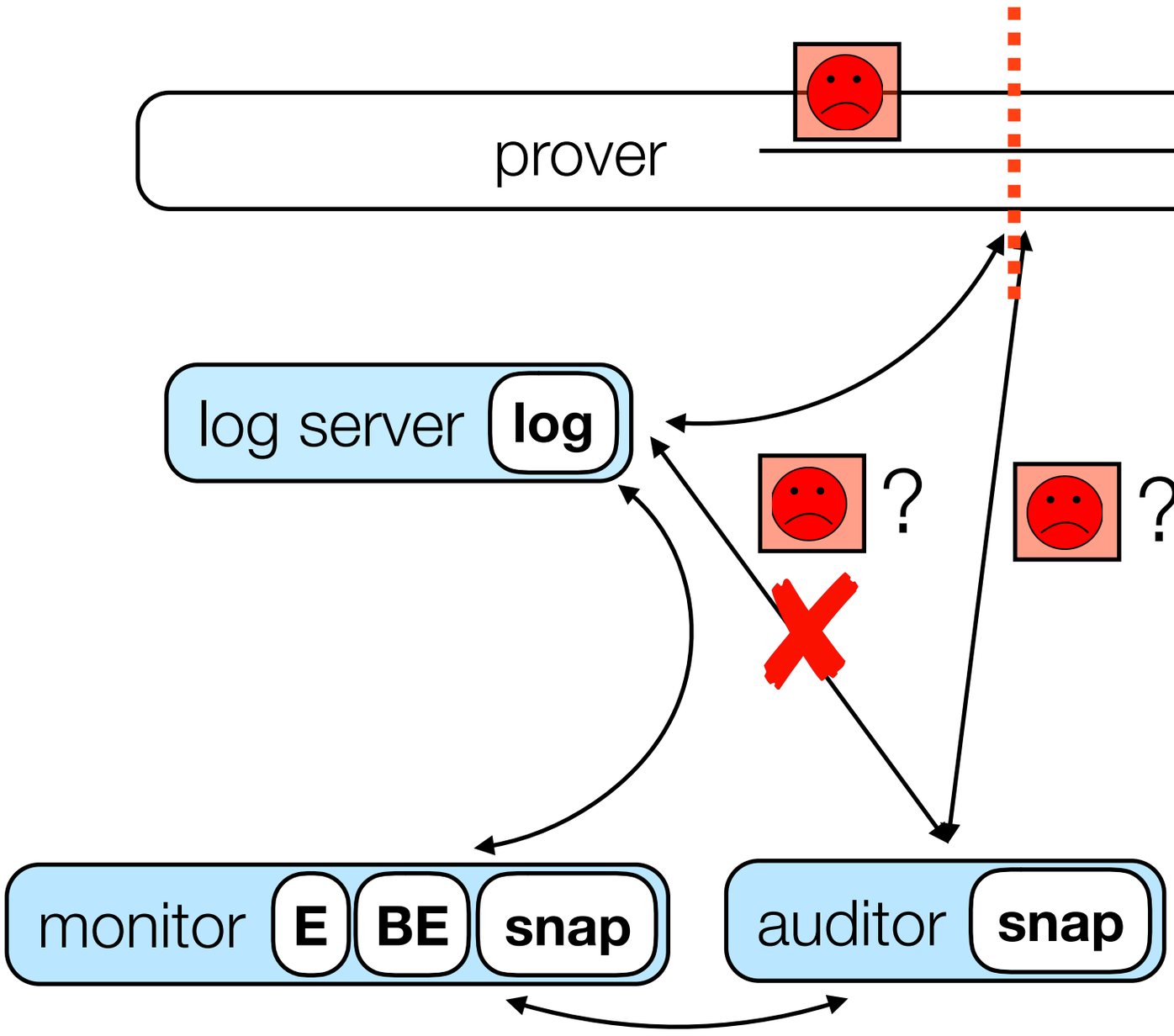
accountability: log server is held to its promises

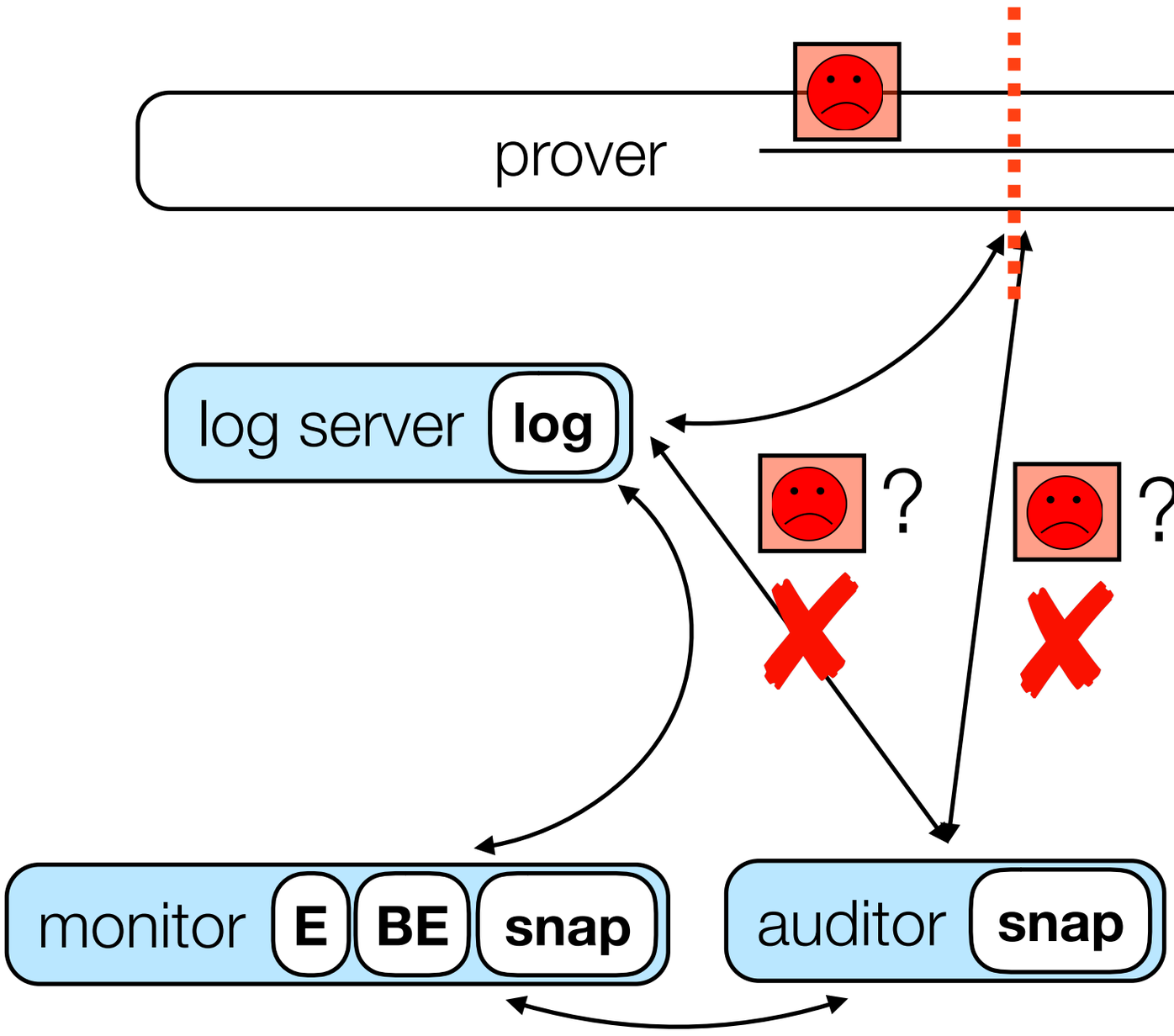

prover
verifier
log server
log
monitor
E
BE
snap
auditor
snap

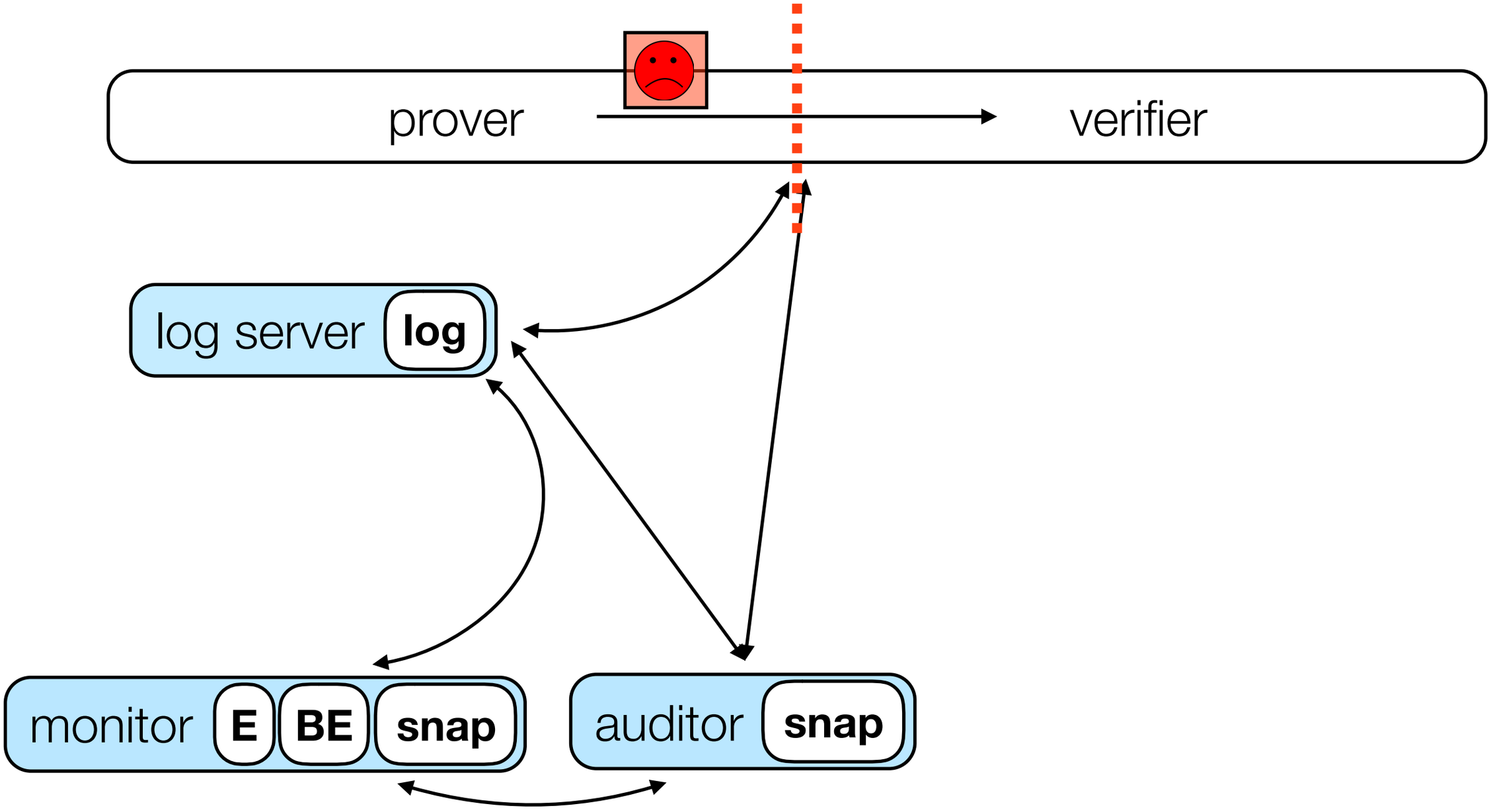

prover
verifier
log server
log
monitor
E
BE
snap
auditor
snap

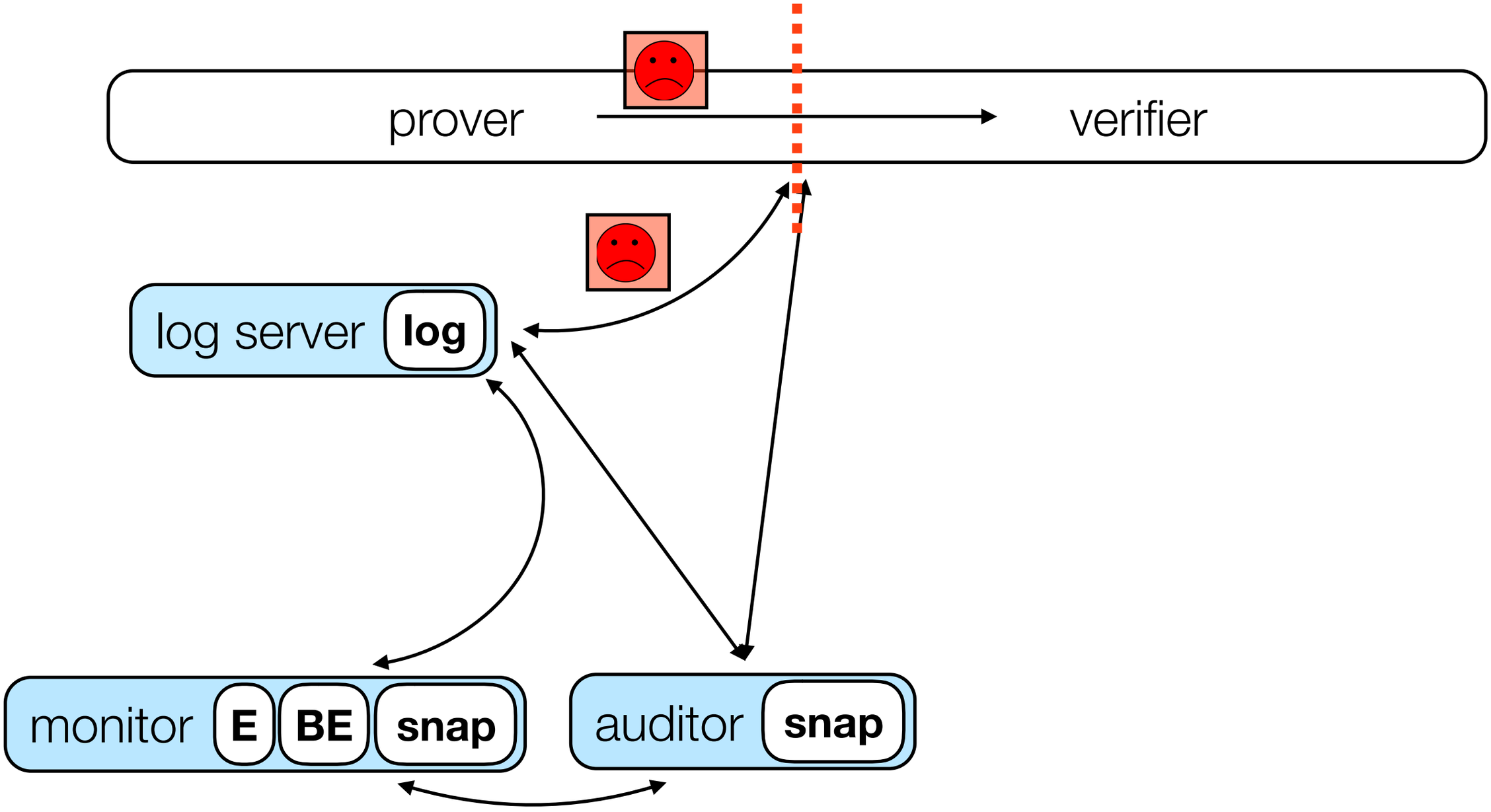

prover
verifier
log server
log
?
monitor
E
BE
snap
auditor
snap

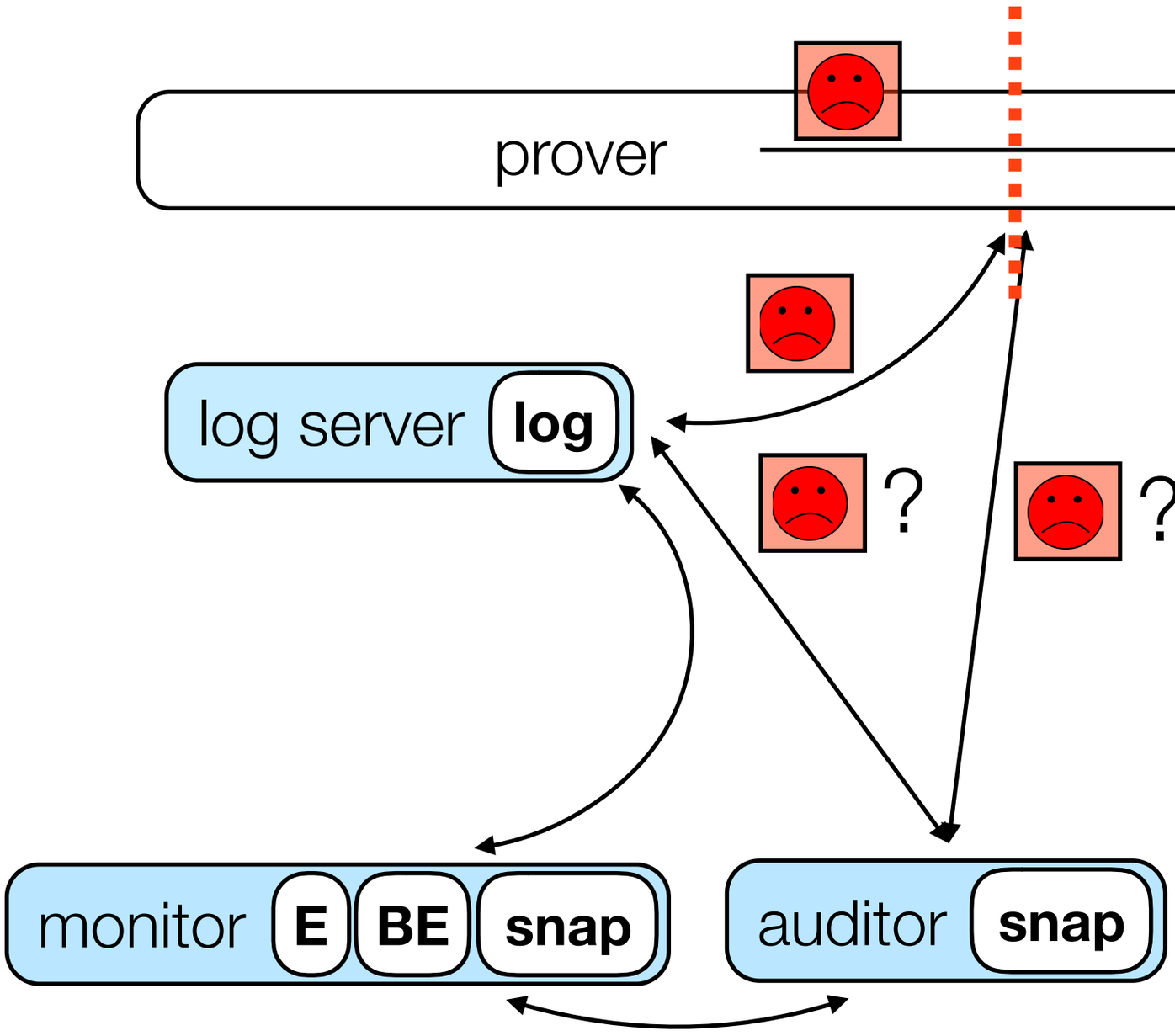

prover
verifier
log server
log
?
?
monitor
E
BE
snap
auditor
snap

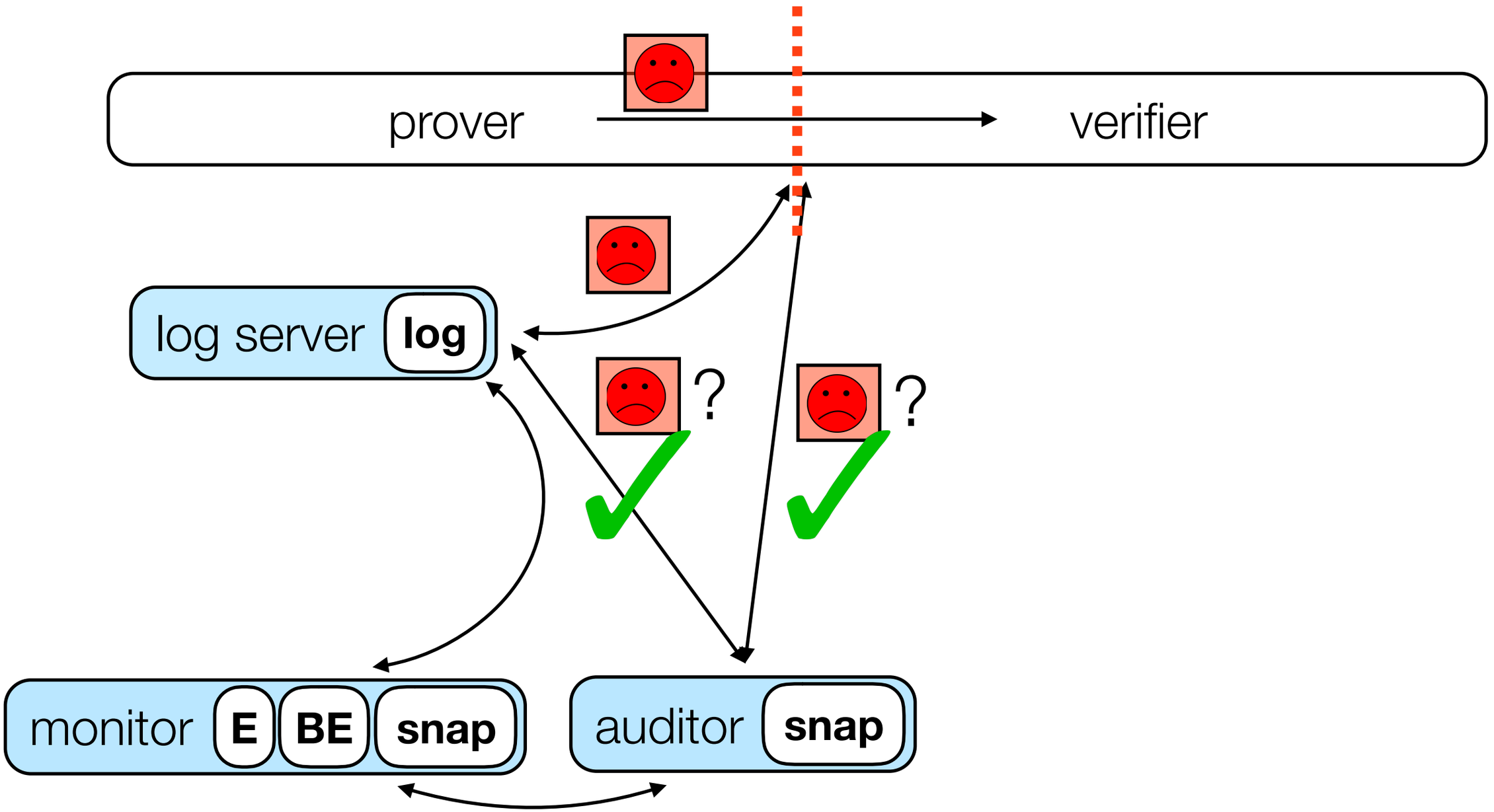

prover
verifier
log server
log
?
?
monitor
E
BE
snap
auditor
snap

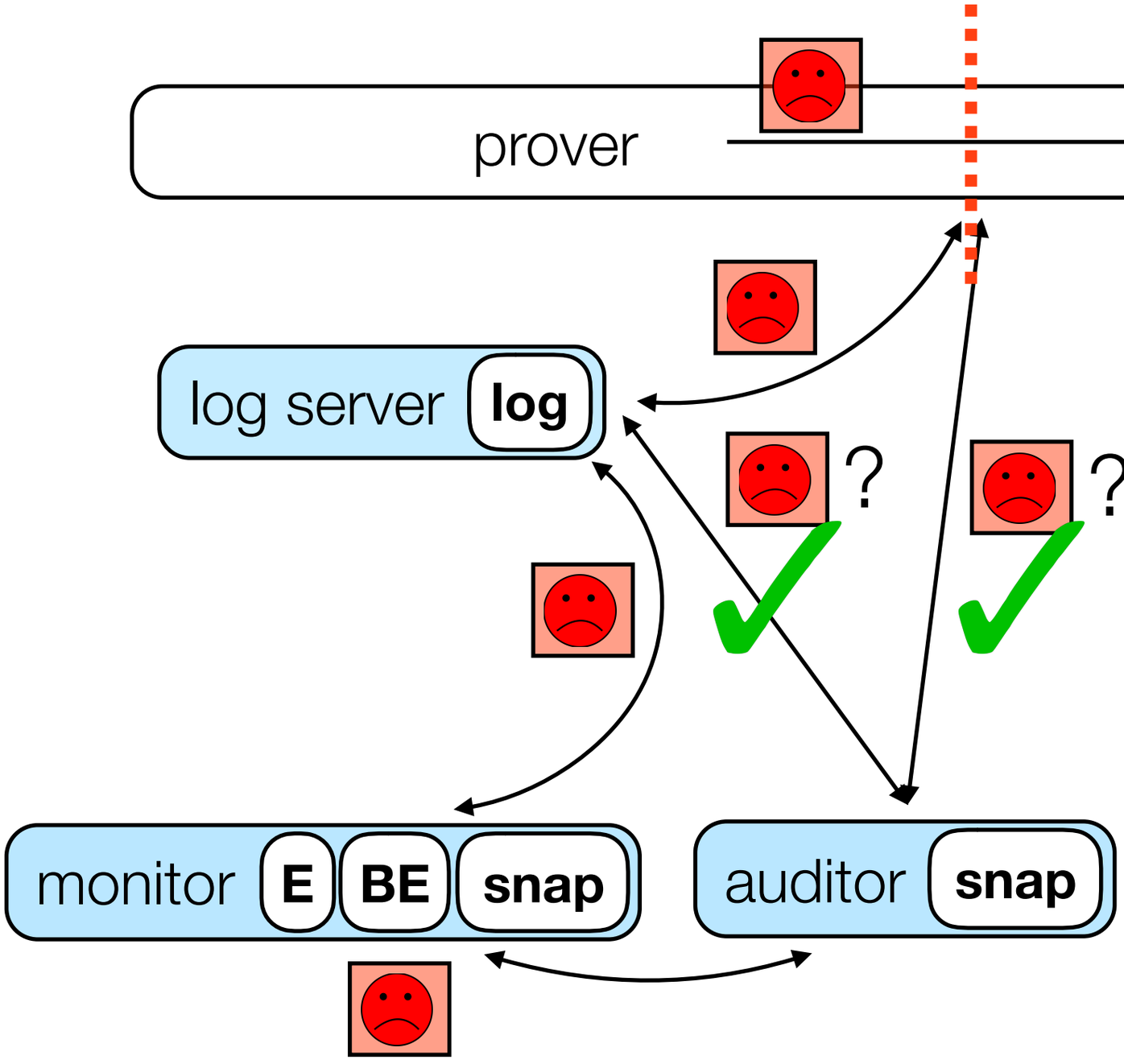

prover
verifier
log server
log
?
?
monitor
E
BE
snap
auditor
snap

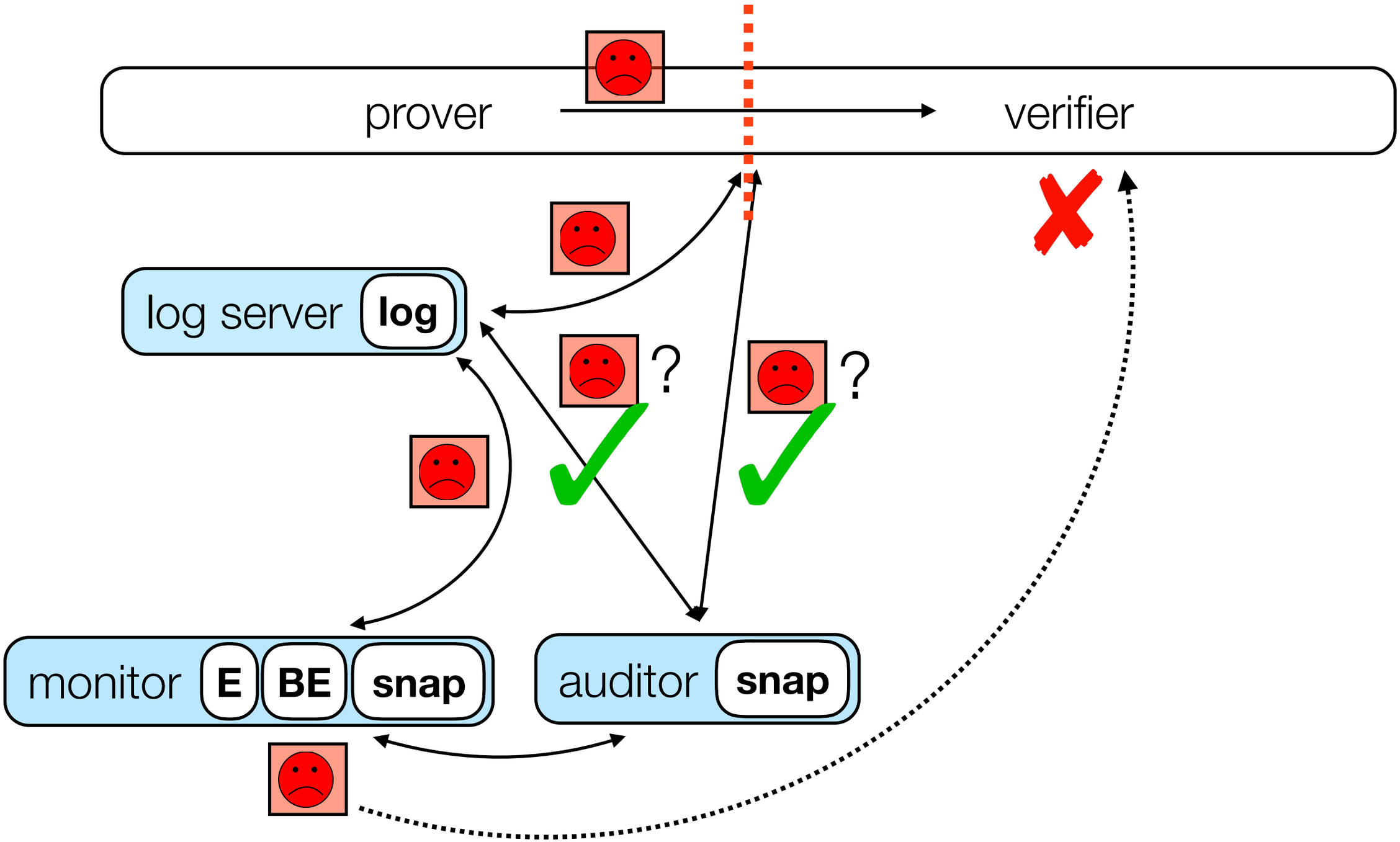

prover
verifier
log server
log
monitor
E
BE
snap
auditor
snap


prover
verifier
log server
log
monitor
E
BE
snap
auditor
snap

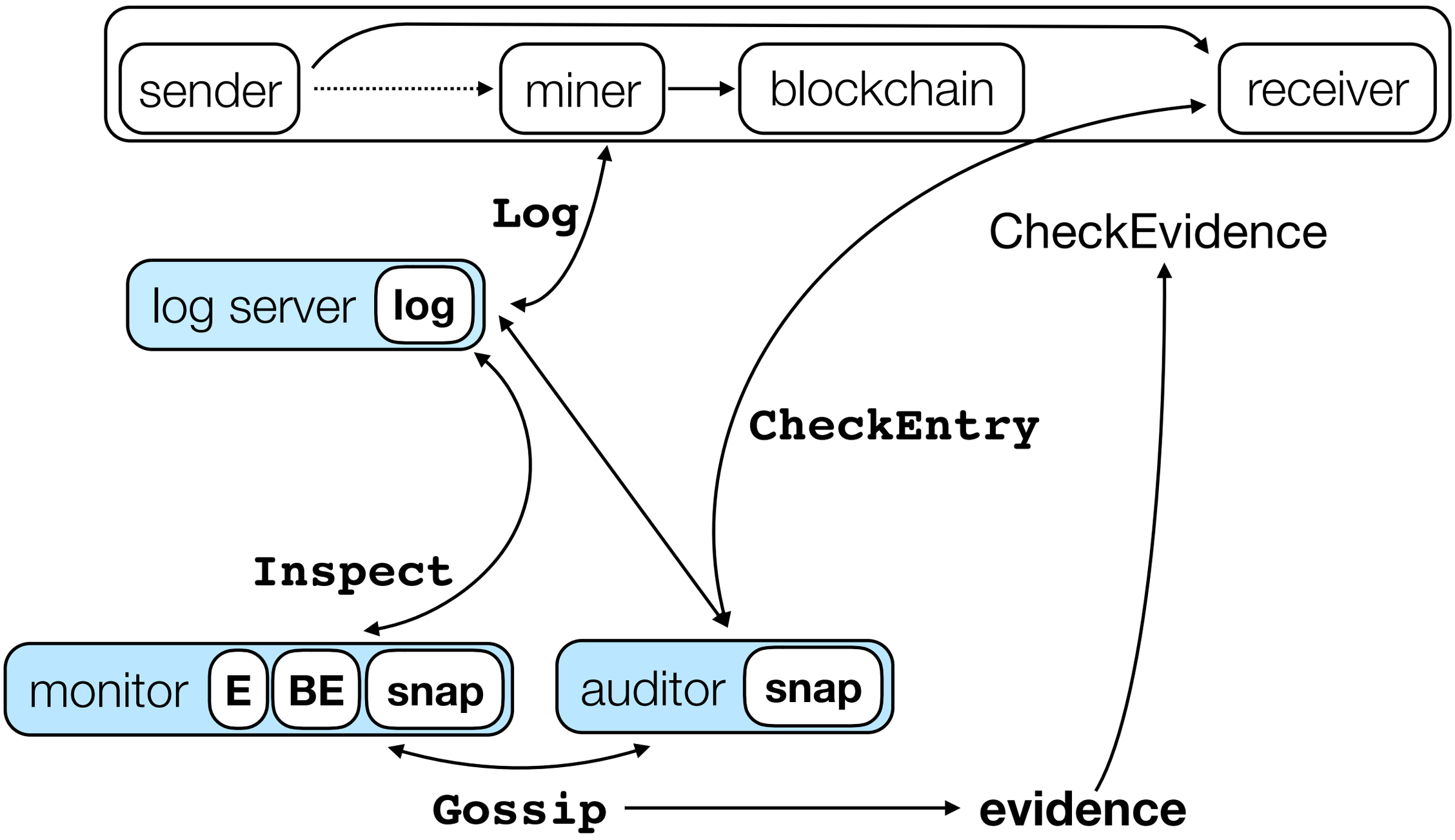

prover
verifier
log server
log
?
?
monitor
E
BE
snap
auditor
snap


prover
verifier
log server
log
?
?
monitor
E
BE
snap
auditor
snap


prover
verifier
log server
log
?
?
monitor
E
BE
snap
auditor
snap

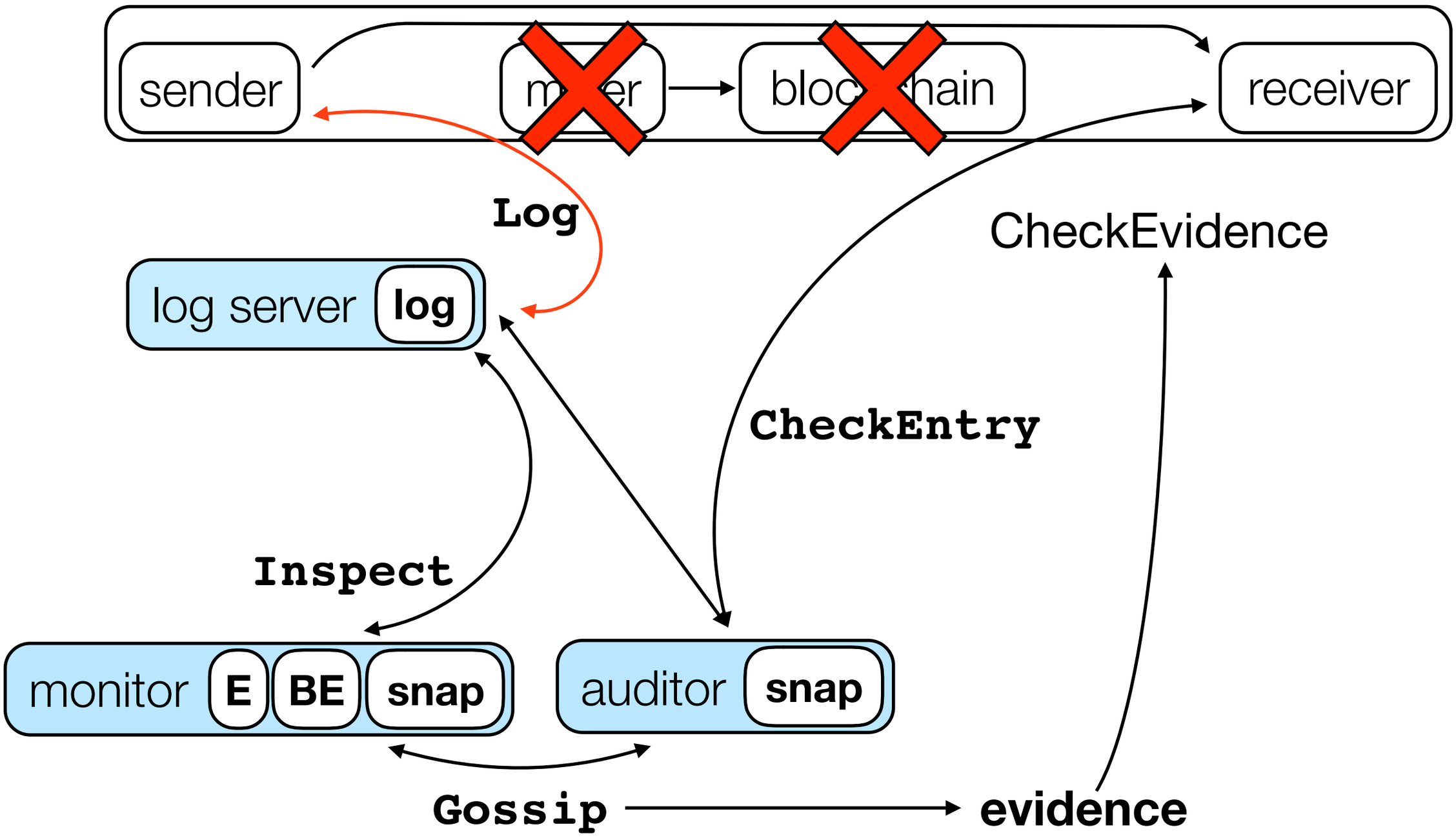

prover
verifier
log server
log
?
?
monitor
E
BE
snap
auditor
snap

Bitcoin



sender and receiver don’t need to store blockchain

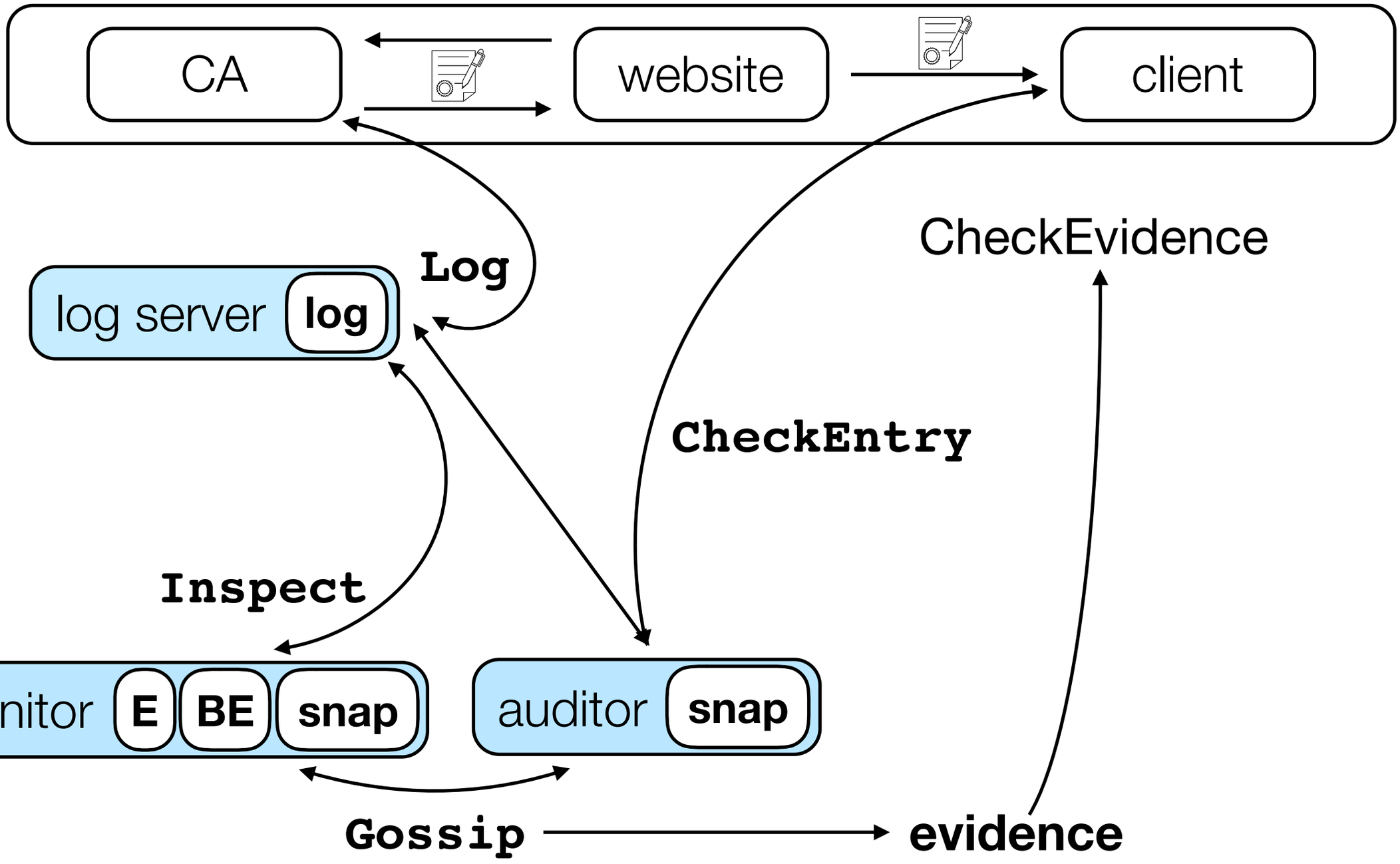
Bitcoin



sender and receiver don’t need to store blockchain

gives rise to hybrid system (≈RSCoin) with no mining

Certificate Transparency [LL13]

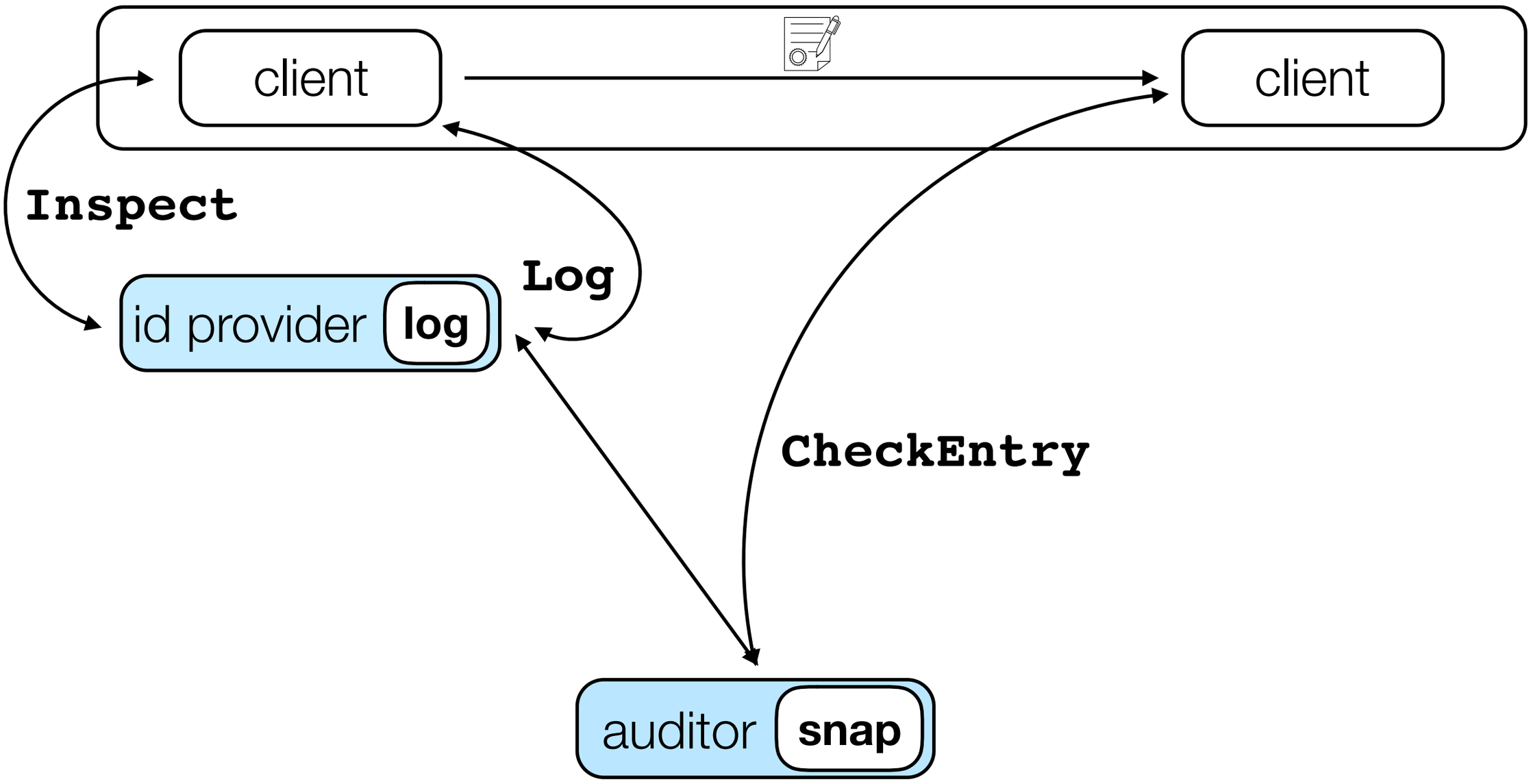


bad certificate issuance is exposed

⇒ clients are less likely to accept bad certificates

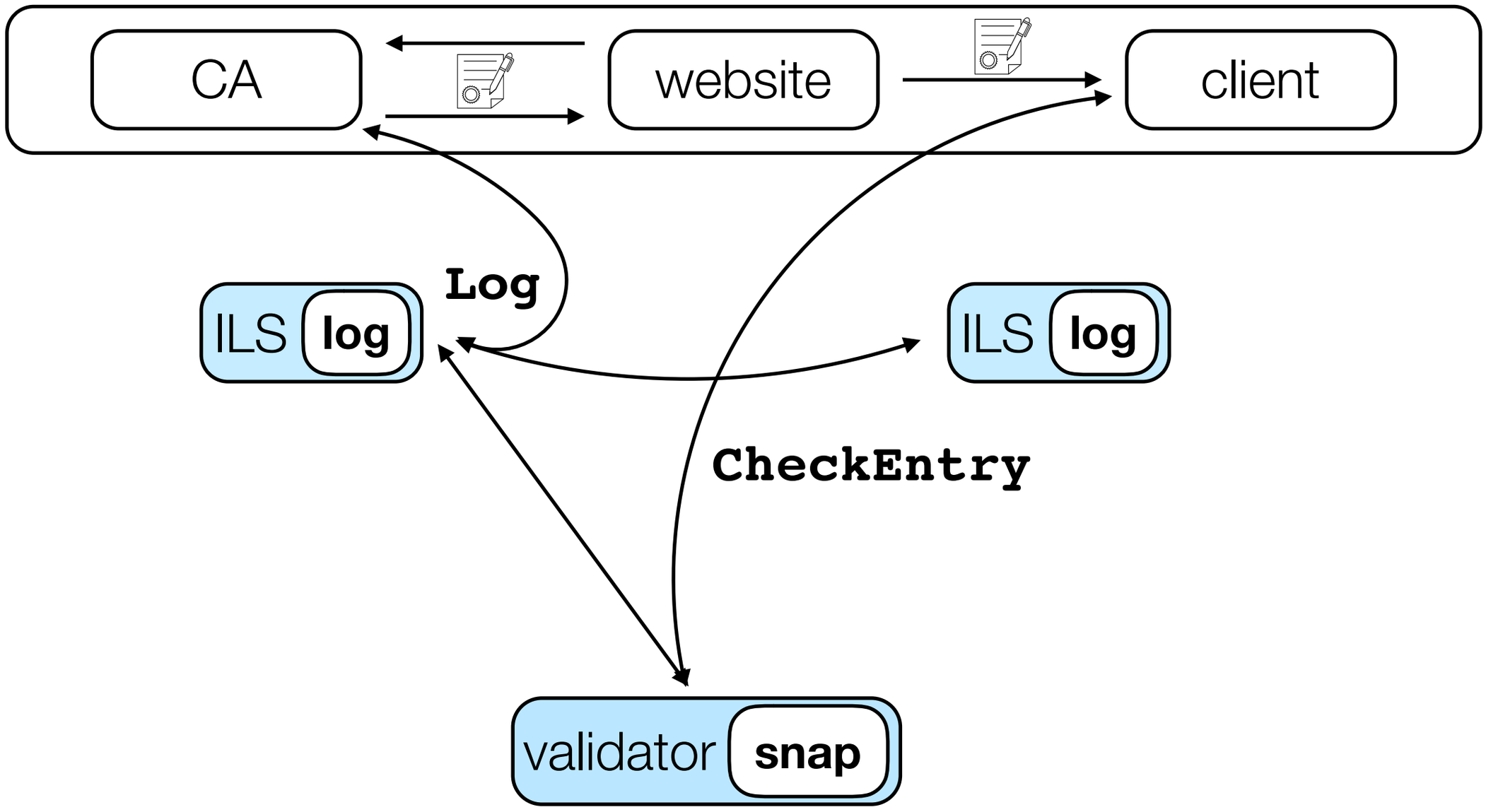
(icon by parkjisun from noun project)

CONIKS [MBBFF USENIX Sec'16]

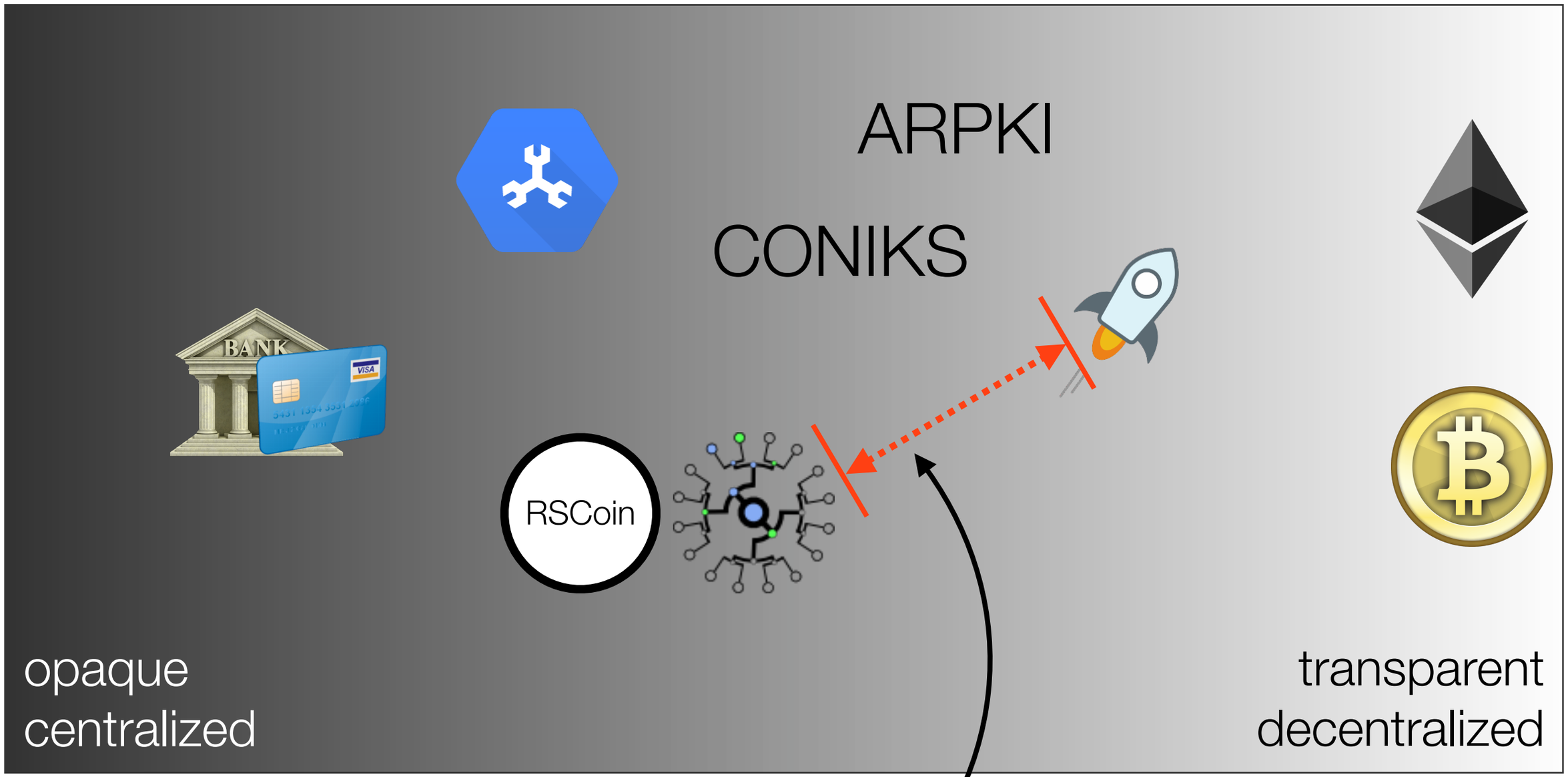


(icon by parkjisun from noun project)

ARPKI [BCKPSS CCS'13]



(icon by parkjisun from noun project)



what is this distance?

security properties

(transparency overlays)

consistency

non-frameability

accountability

security properties

(transparency overlays)

consistency

non-frameability

accountability

(RSCoin)

no double spending

non-repudiation

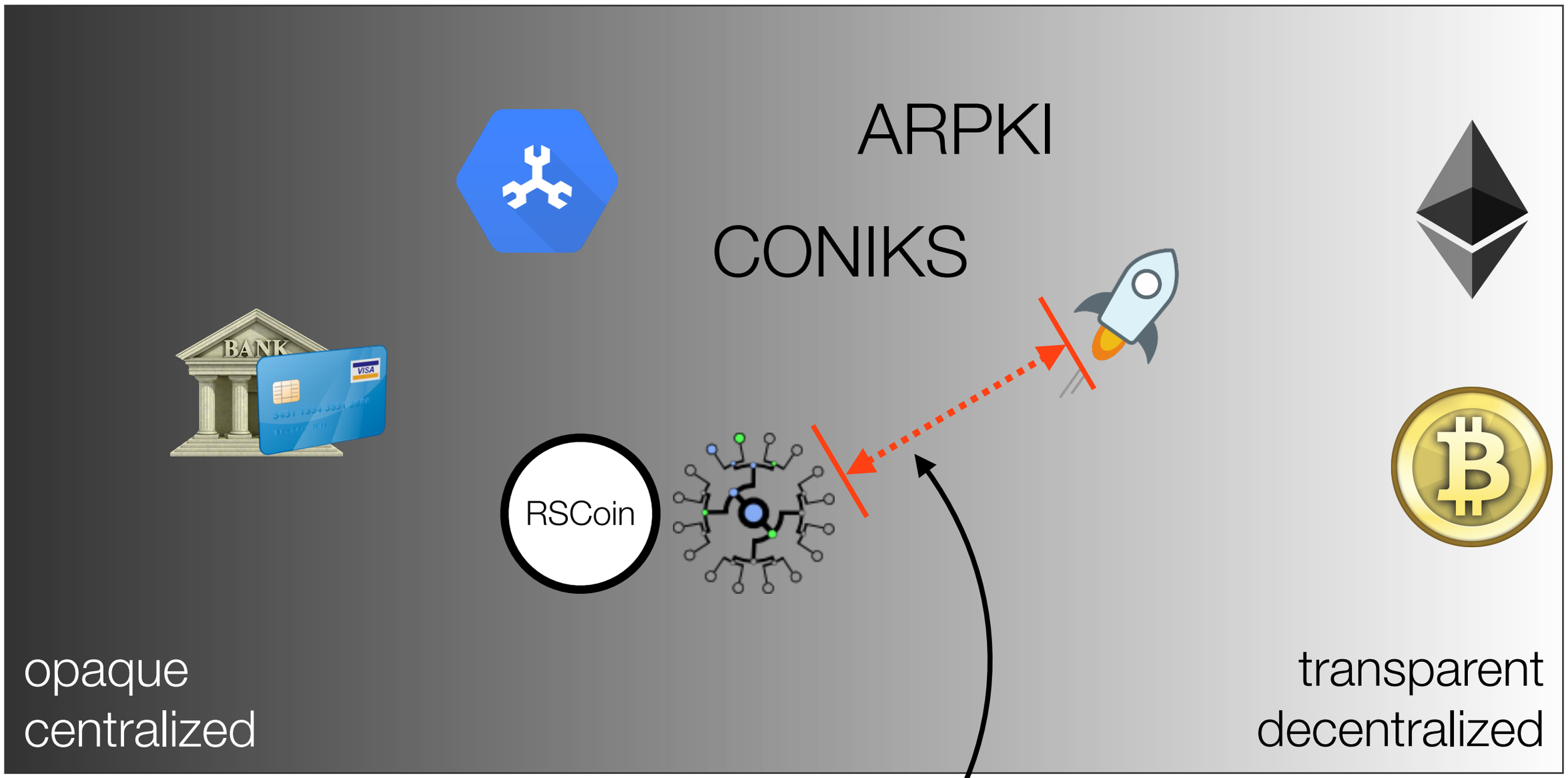
auditability

security properties

(transparency overlays)		(RSCoin)
consistency	⇔	no double spending
non-frameability	⇖⇘	non-repudiation
accountability	⇔	auditability

security properties

(transparency overlays)		(RSCoin)
consistency	⇔	no double spending
non-frameability	⇎	non-repudiation
accountability	⇔	auditability
privacy (of what)?		privacy (of what)?



what is this distance?

what security properties to look for?

10 usability

9 governance

8 comparisons

7 key management

6 agility

5 interoperability

4 scalability

3 cost-effectiveness

2 privacy

1 scalability

10 usability

9 governance

8 comparisons

7 key management

6 agility

5 interoperability

4 scalability

3 cost-effectiveness

2 privacy

1 scalability

Thanks! Any questions?