Snappy
Fast On-chain Payments with Practical Collaterals

Vasilios Mavroudis[*], Karl Wüst[†], Aritra Dhar[†]
Kari Kostiainen[†], Srdjan Capkun[†]

[*]*University College London* [†]*ETH Zurich*

Cryptocurrencies based on permissionless blockchains could

- Decentralize the global financial system
- Reduce trust assumptions
- Increase operational transparency
- Improve user privacy

Open Challenges

	Centralized Processors	Permissionless Blockchains
Throughput	Thousands of txs/sec	Tenths of txs/sec
Latency	Confirmation in <3 sec	Minutes to finality
Privacy	Trusted third party needed	[0, full privacy)

Open Challenges

	Centralized Processors	Permissionless Blockchains
Throughput	Thousands of txs/sec	Tenths of txs/sec
Latency	Confirmation in <3 sec	Minutes to finality
Privacy	Trusted third party needed	[0, full privacy)

- *Retail Payments*
- *Point-of-Sale Purchases*
- *Time-critical Transactions*

On-chain Improvements

- e.g., Proof-of-Stake, Sharding
- Improve the throughput of the blockchain.
- Improve latency only under a relaxed threat model.

On-chain Improvements

- e.g., Proof-of-Stake, Sharding
- Improve the throughput of the blockchain.
- Improve latency only under a relaxed threat model.

No improvement in latency under the original threat model.

Layer 2 Protocols

- Move transactions off the chain.
- Use the blockchain only when necessary.
- High-throughput and low-latency.

Layer 2 Protocols

- Move transactions off the chain.
- Use the blockchain only when necessary.
- High-throughput and low-latency.

Payment channels

- Large amount of locked-in funds for customers.
- Require a separate deposit for each channel.
- Pre-deposit their future expenditure.

Layer 2 Protocols

- Move transactions off the chain.
- Use the blockchain only when necessary.
- High-throughput and low-latency.

Payment channels

- Large amount of locked-in funds for customers.
- Require a separate deposit for each channel.
- Pre-deposit their future expenditure.

Payment networks, **Payment hubs, Side-chains**

- Incompatible with the unilateral nature of retail payments (no rebalancing).
- Additional trust assumptions.

Snappy

- Low latency (<2 secs) suitable for retail payments.
- Operates on top of low-throughput and high-latency blockchains.
- Future on top of high-throughput and high/mid-latency blockchains.

Snappy

- Low latency (<2 secs) suitable for retail payments.
- Operates on top of low-throughput and high-latency blockchains.
- Future on top of high-throughput and high/mid-latency blockchains.

Key Features

- No changes to the underlying consensus protocol. ✔
- No additional trust assumptions. ✔
- No additional operational requirements. ✔

Snappy

- Low latency (<2 secs) suitable for retail payments.
- Operates on top of low-throughput and high-latency blockchains.
- Future on top of high-throughput and high/mid-latency blockchains.

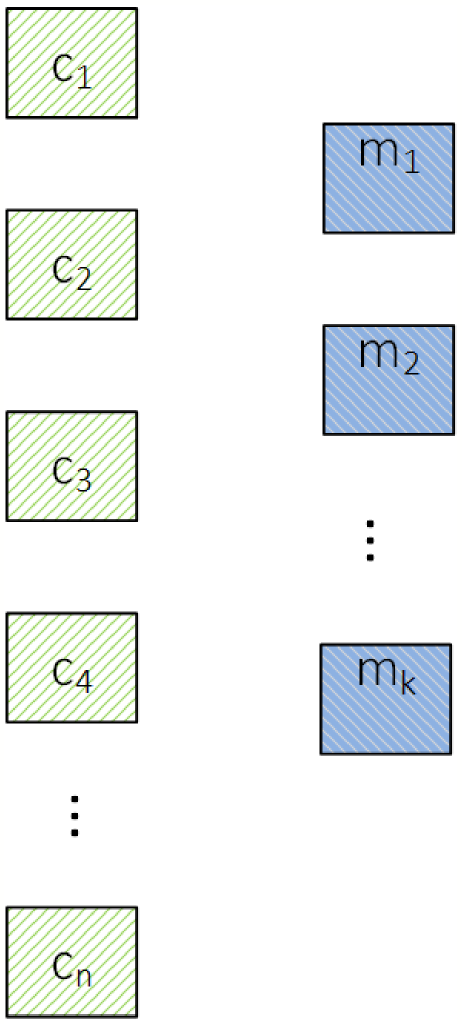
Key Features

- No changes to the underlying consensus protocol. ✓
- No additional trust assumptions. ✓
- No additional operational requirements. ✓
- Small opportunity cost.
- Requires smartcontract language.

Snappy

Application scenarios

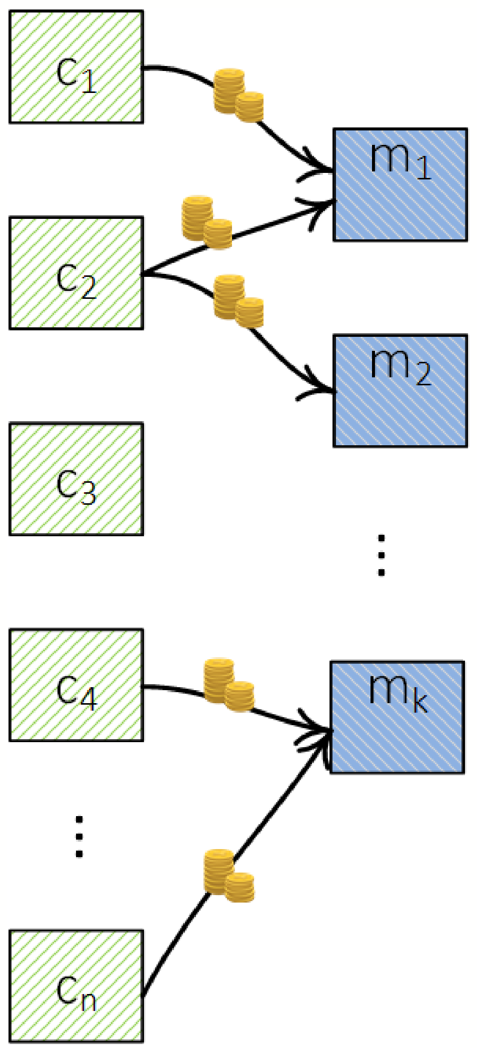
- ❖ A large number of users (e.g., 1,000,000 customers).
- ❖ A moderate set of recipients (e.g., 100 merchants).



Snappy

Application scenarios

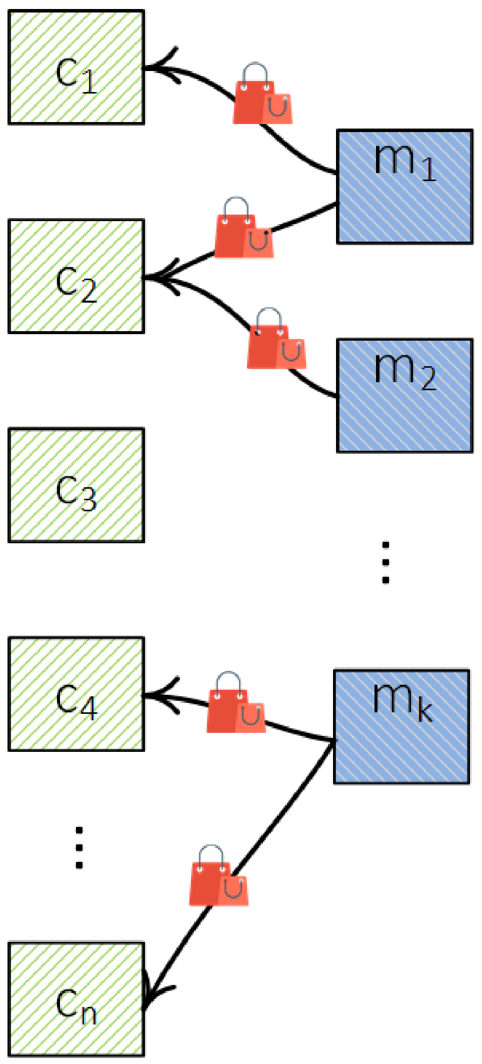
- ❖ A large number of users (e.g., 100,000 customers).
- ❖ A moderate set of recipients (e.g., 100 merchants).
- ❖ Users pay the recipients.
- ❖ Small- to mid-value transactions.



Snappy

Application scenarios

- ❖ A large number of users (e.g., 100,000 customers).
- ❖ A moderate set of recipients (e.g., 100 merchants).
- ❖ Users pay the recipients.
- ❖ Small- to mid-value transactions.
- ❖ The recipients give the products, once they receive the funds.



How does latency occur?

- Block interval (e.g., ~13 seconds for Ethereum)
- Probabilistic finality (>1 confirmations)
- The number of confirmations,
  depends on the transaction value

How does latency occur?

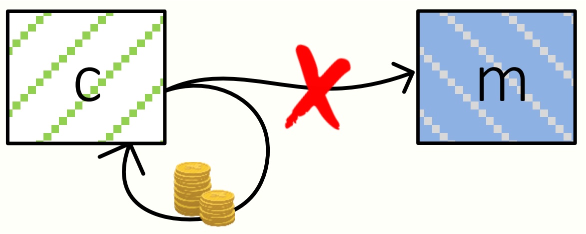
- Block interval (e.g., ~13 seconds for Ethereum)
- Probabilistic finality (>1 confirmations)
- The number of confirmations, depends on the transaction value

Can we do zero-confirmation txs?

How does latency occur?

- Block interval (e.g., ~13 seconds for Ethereum)
- Probabilistic finality (>1 confirmations)
- The number of confirmations, depends on the transaction value

Can we do zero-confirmation txs?



Trivial Solutions

- Convince your supermarket to trust you?
- Pre-deposit funds to your local supermarket?
- Try to catch double-spending early?

Trivial Solutions

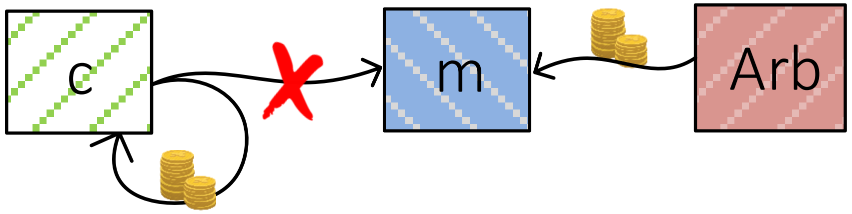
- Convince your supermarket to trust you?
- Pre-deposit funds to your local supermarket?
- Try to catch double-spending early?

Can we do better?

- Customers keep their money in their wallet.
- Merchants guaranteed to get their money.
- No trust to/reliance on third parties.

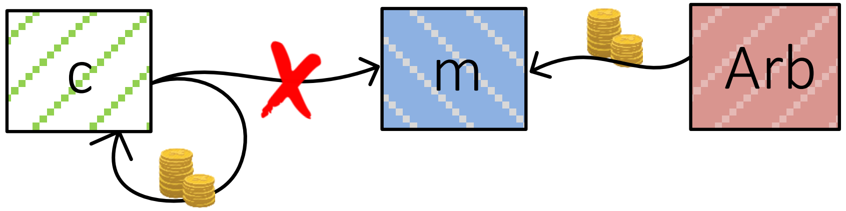
Idea: Collaterals

1. Customer places collateral (e.g., $100) on a smartcontract.
2. Victim merchants can claim funds if the customer cheats.



Idea: Collaterals

1. Customer places collateral (e.g., \$100) on a smartcontract.
2. Victim merchants can claim funds if the customer cheats.

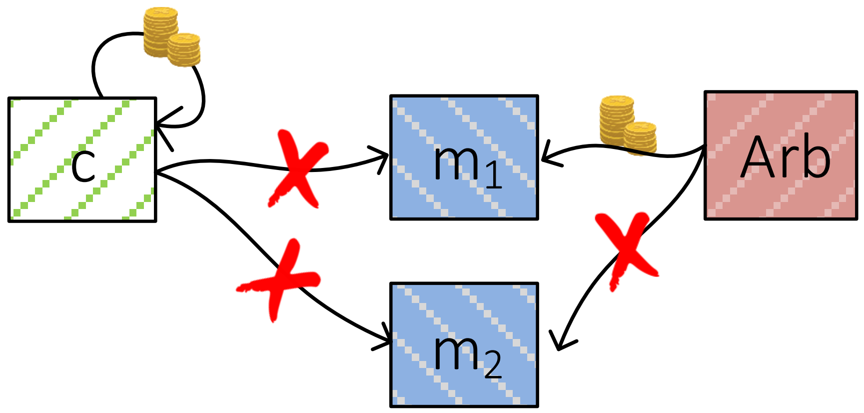


A settlement "claim" requires

- The payment transaction (given to the merchant by the customer).
- Its conflicting transaction (from the blockchain).
- In Ethereum, conflicting transactions share the same *nonce*.

The collateral is used only when doublespending!

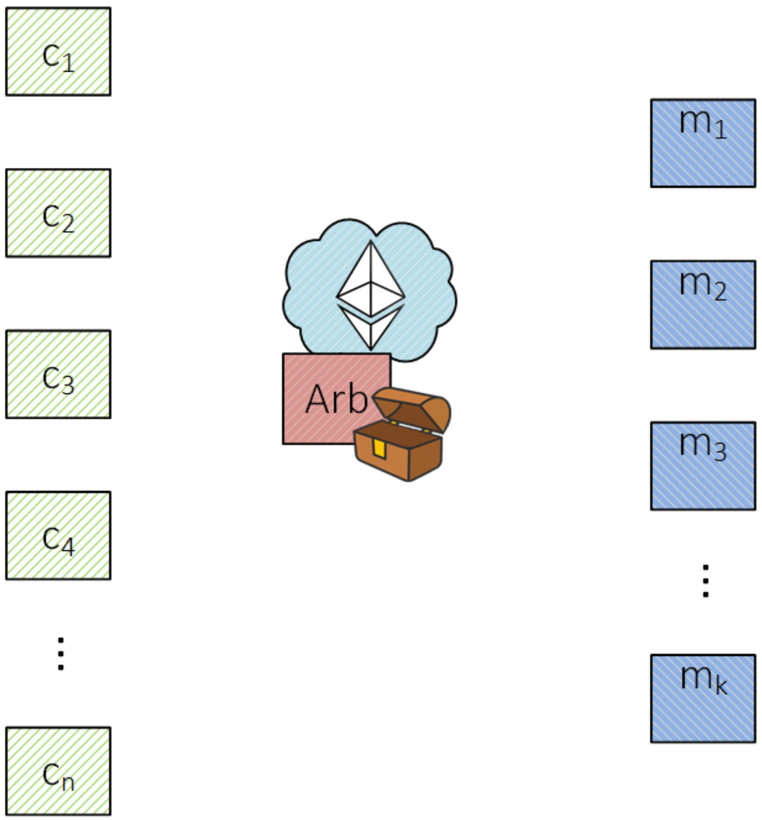
Triple-spending Attack



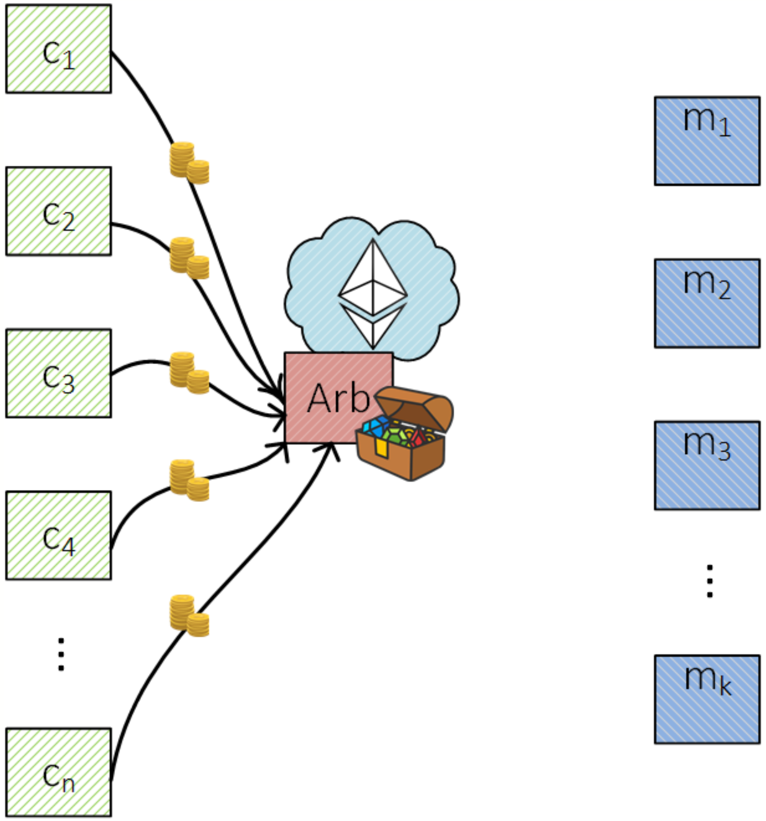
Scaling collaterals to multiple merchants

- Need to keep track of “pending” transactions.
- Merchants accept payment, if the collateral suffices for everyone.

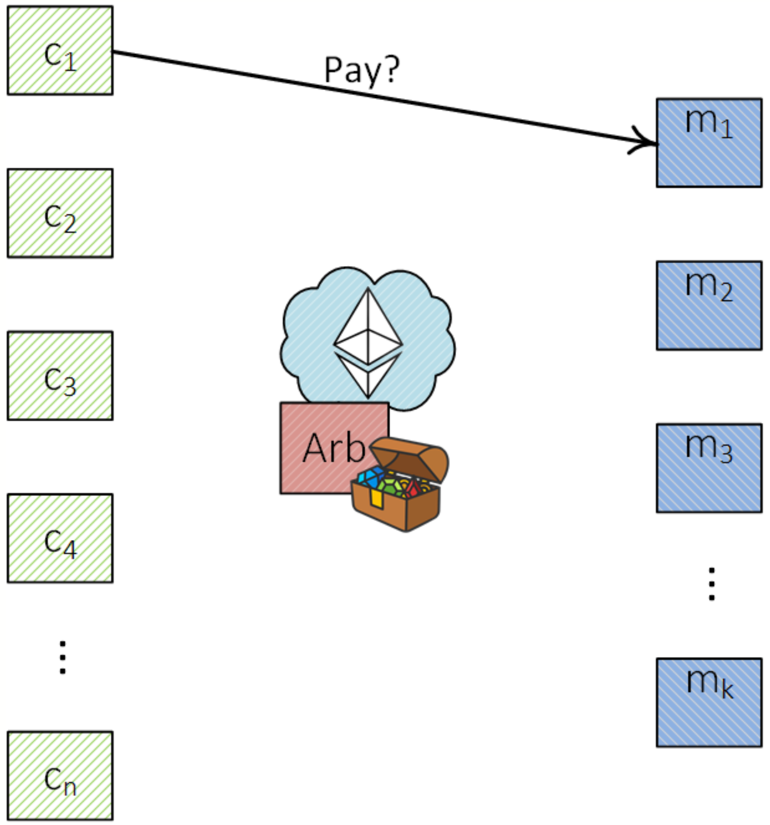
Proposal #1: Trusted Merchants



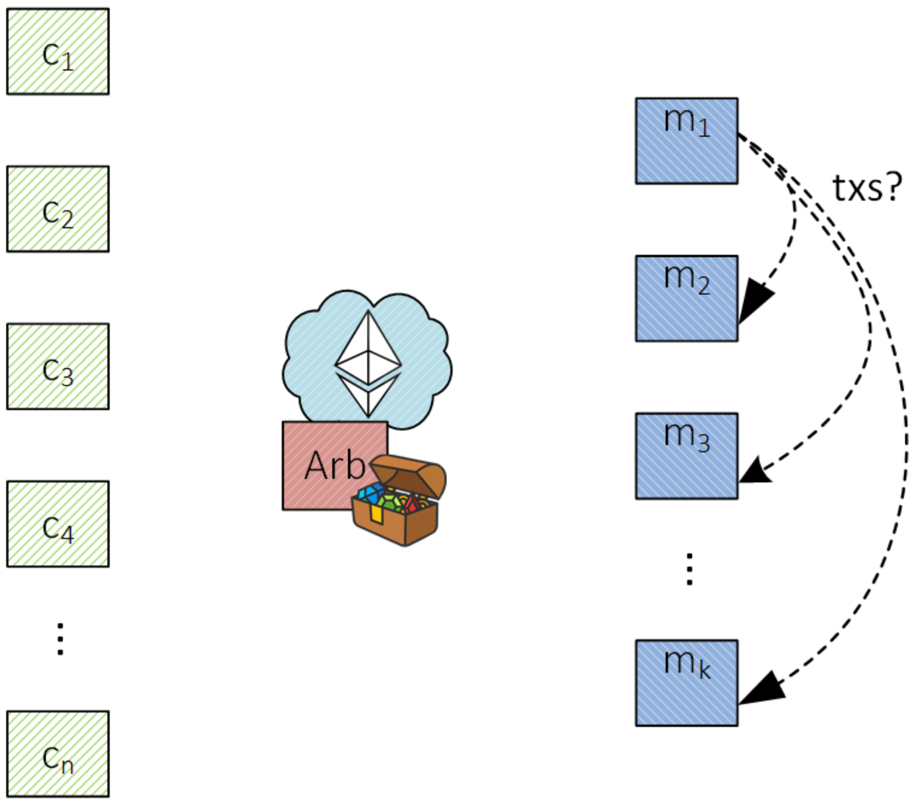
Proposal #1: Trusted Merchants



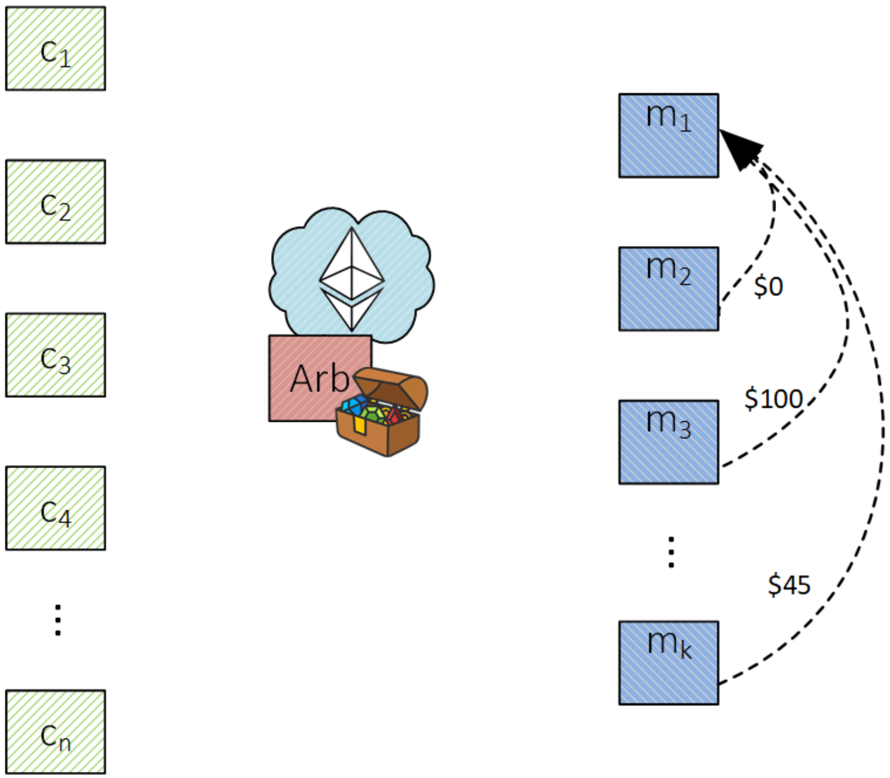
Proposal #1: Trusted Merchants



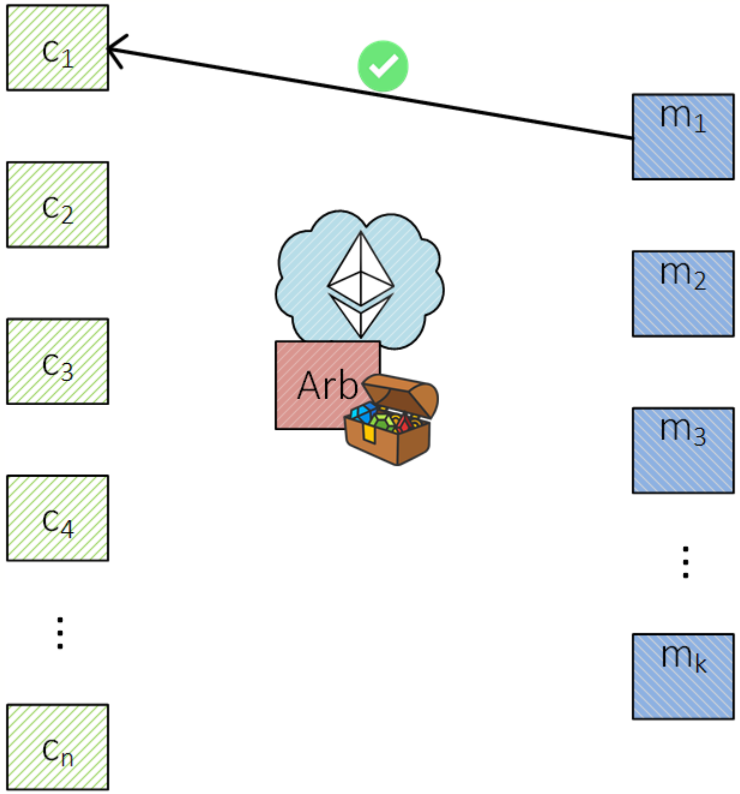
Proposal #1: Trusted Merchants



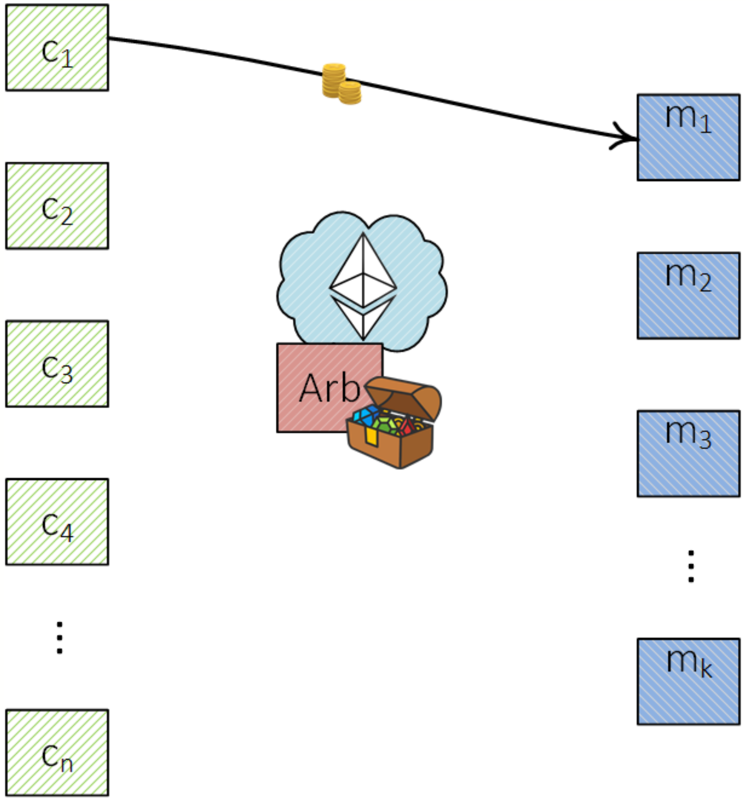
Proposal #1: Trusted Merchants



Proposal #1: Trusted Merchants



Proposal #1: Trusted Merchants



Proposal #1: Trusted Merchants

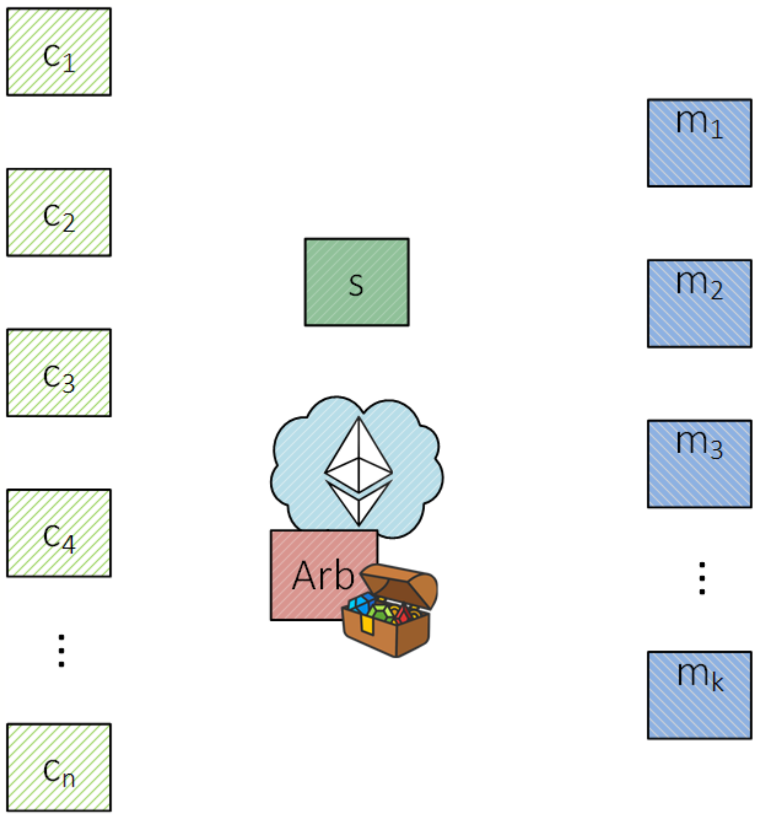
Drawbacks

- ❖ Assumes all merchants are trustworthy.
- ❖ Requires 100% availability of all merchants.

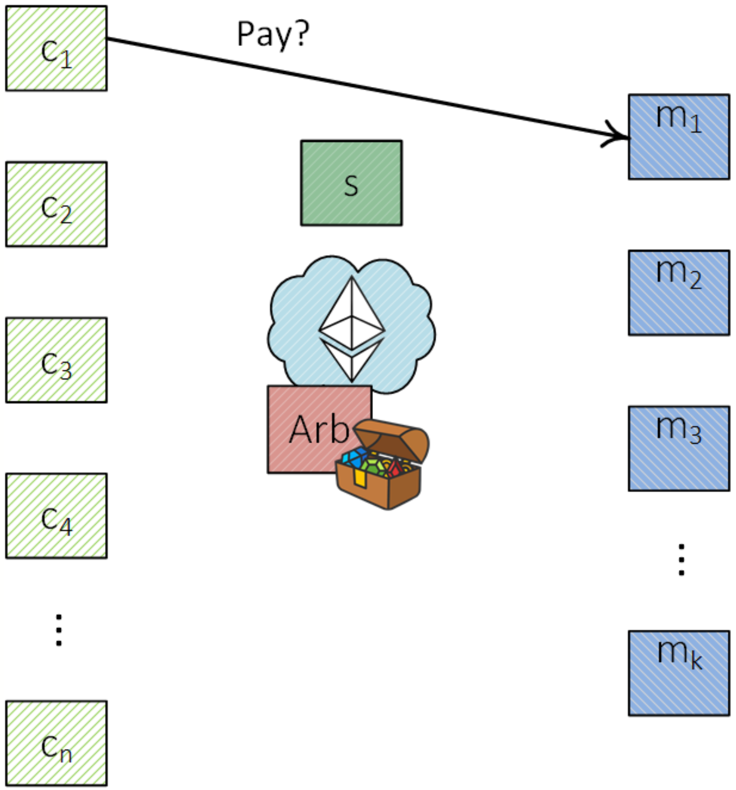
Side-chain variant

- ❖ Additional trust assumptions
- ❖ e.g., BFT -> 1/3 malicious merchants

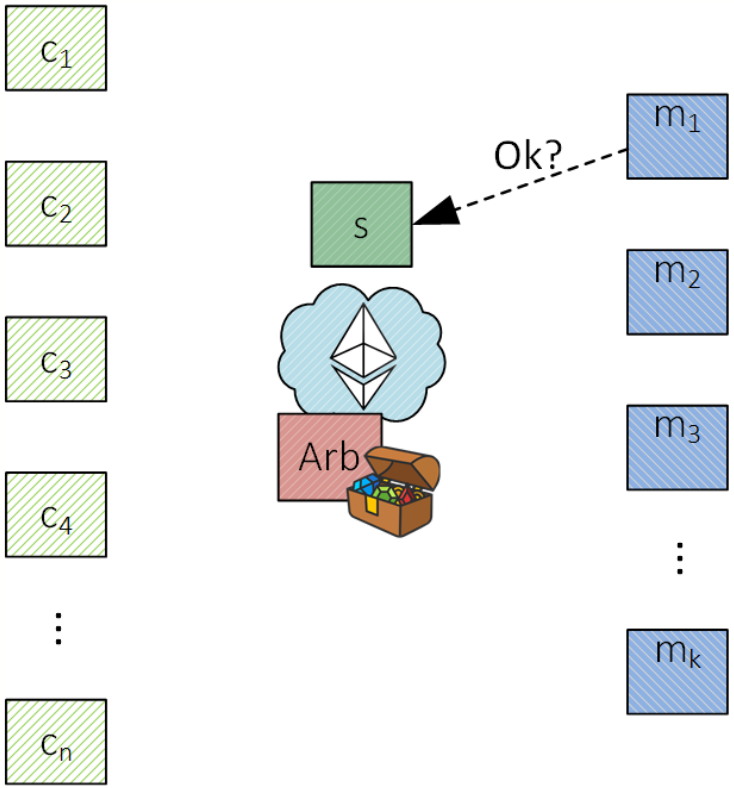
Proposal #2: Trusted Third Party



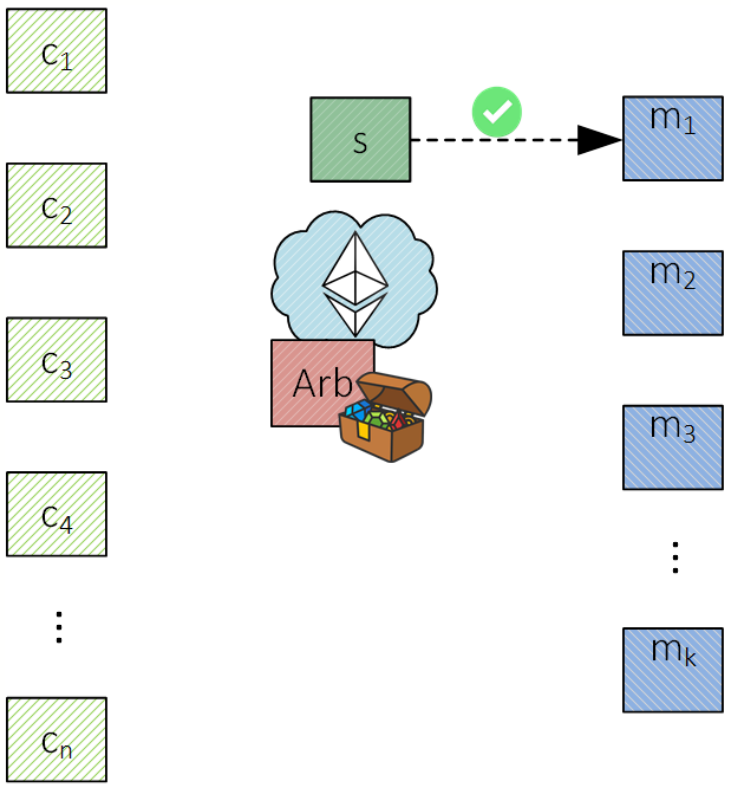
Proposal #2: Trusted Third Party



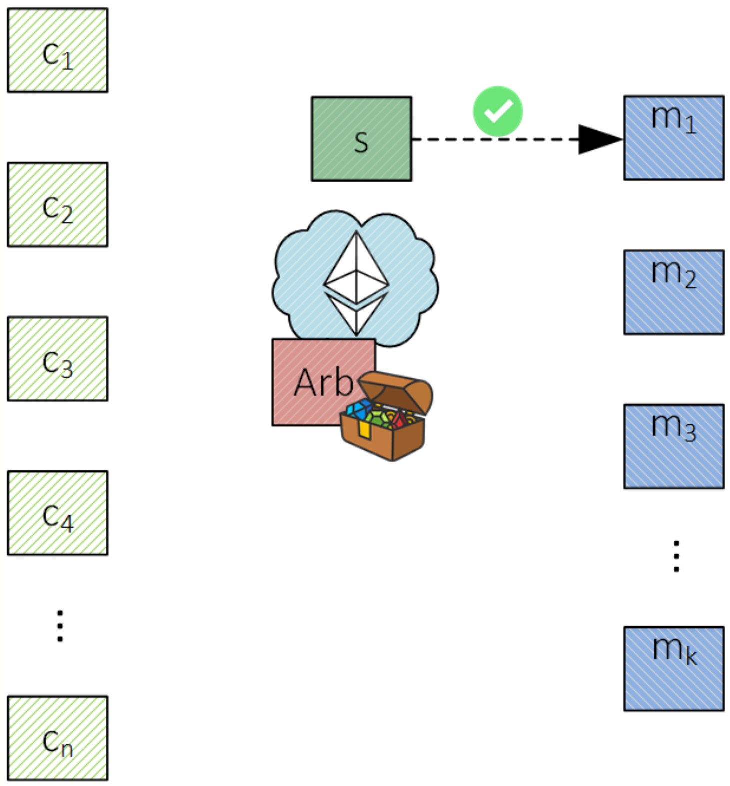
Proposal #2: Trusted Third Party



Proposal #2: Trusted Third Party



Proposal #2: Trusted Third Party

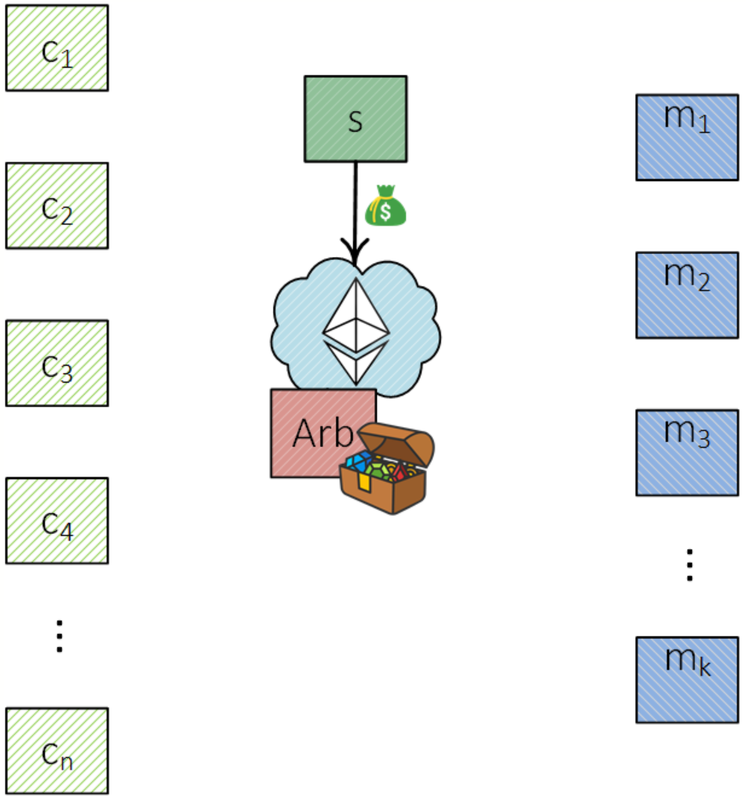


Drawbacks

- What if the statekeeper equivocates?
- What if the statekeeper colludes with customers?

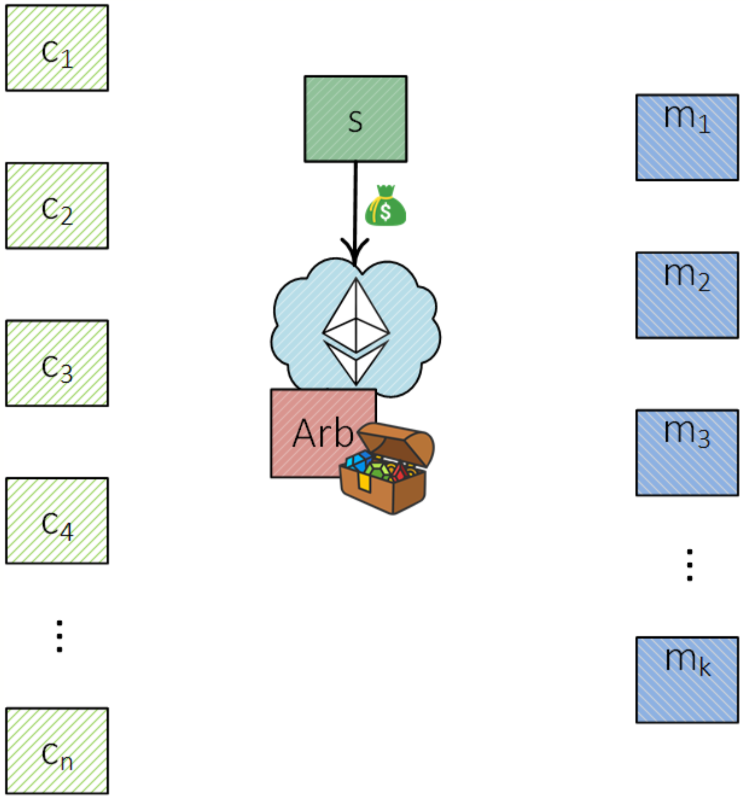
Proposal #3: Untrusted Third Party

- Almost the same as before
- Statekeeper places collateral per merchant.
- If the customer's collateral get depleted, the statekeeper's collateral is used.



Proposal #3: Untrusted Third Party

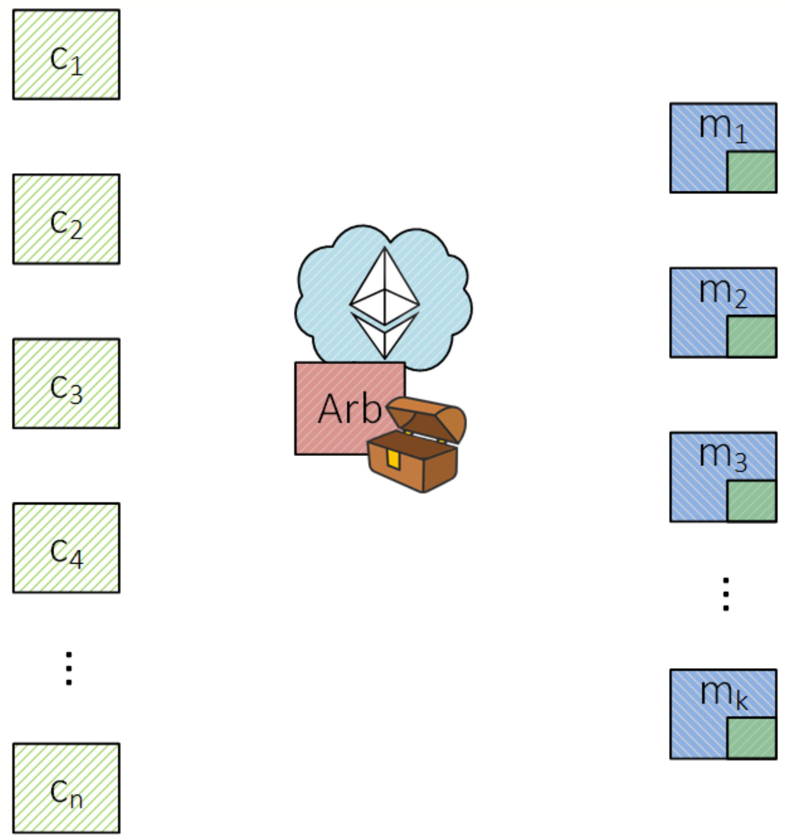
- Almost the same as before
- Statekeeper places collateral <u>per merchant.</u>
- If the customer's collateral get depleted, the statekeeper's collateral is used.



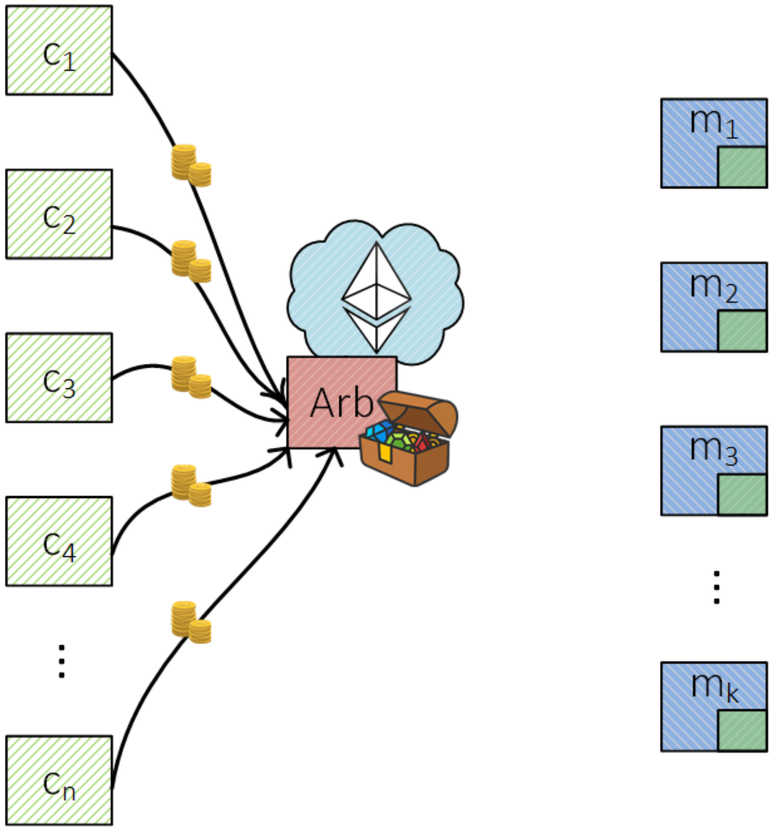
Drawbacks

- We still rely on a third party.

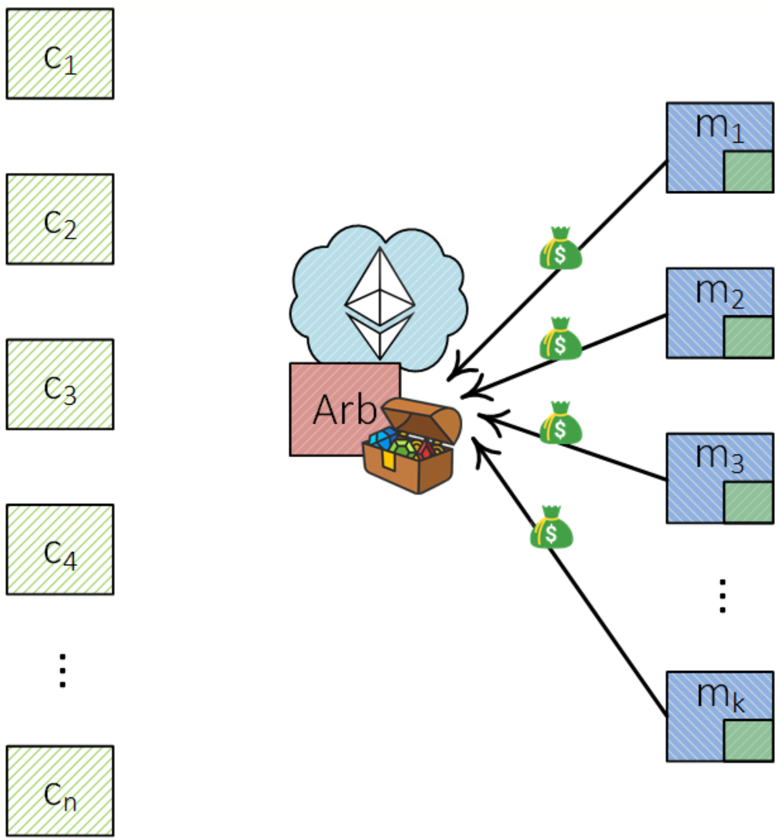
Snappy: Statekeeping Merchants



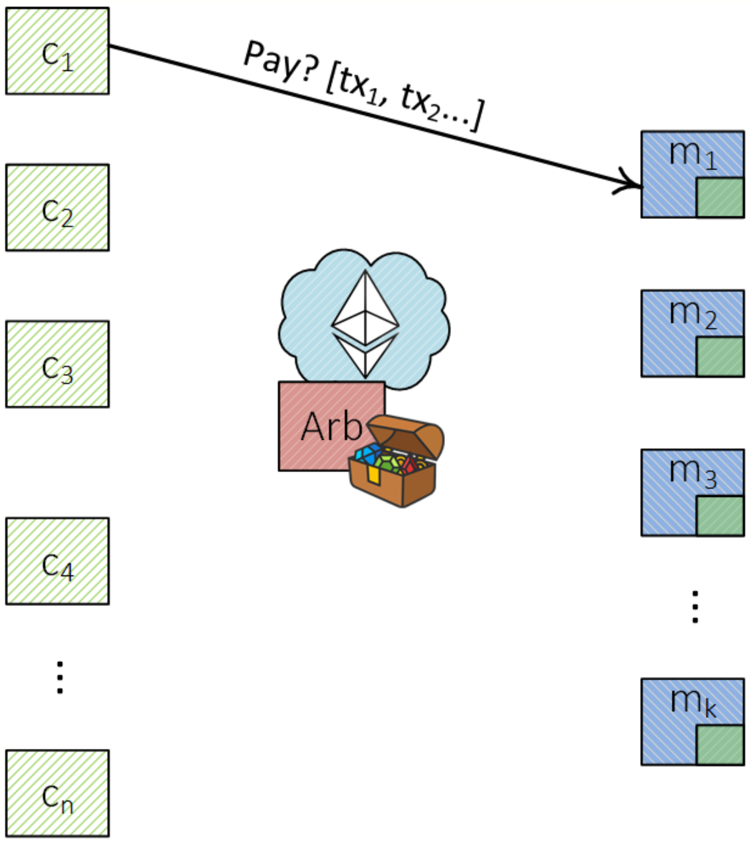
Snappy: Statekeeping Merchants



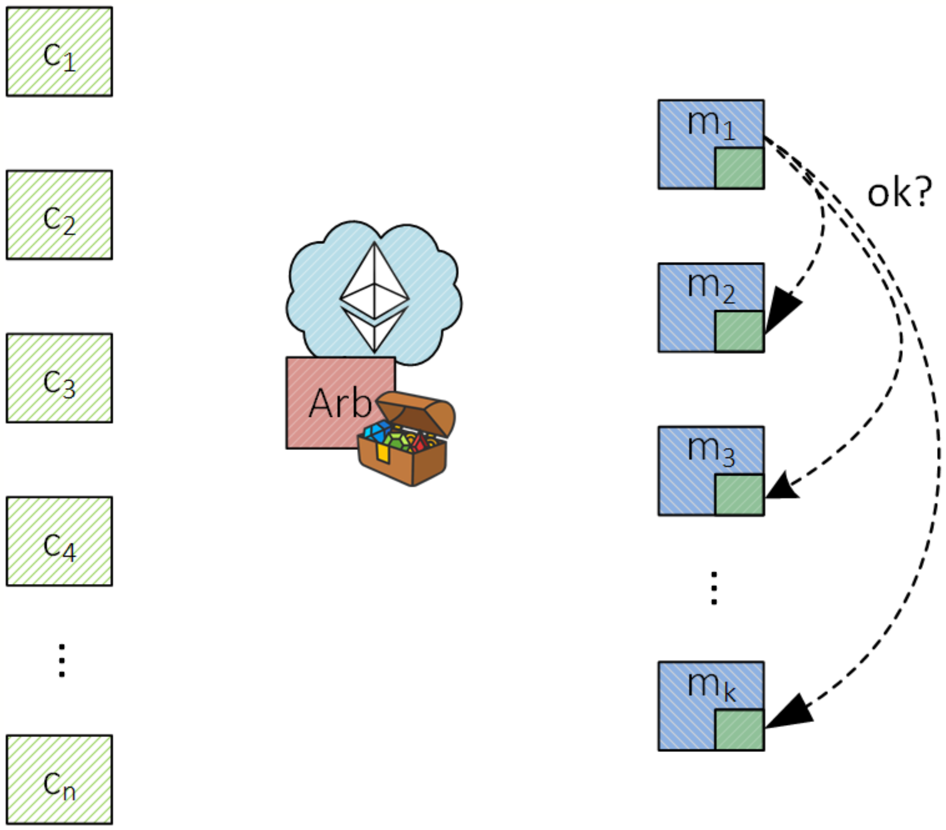
Snappy: Statekeeping Merchants



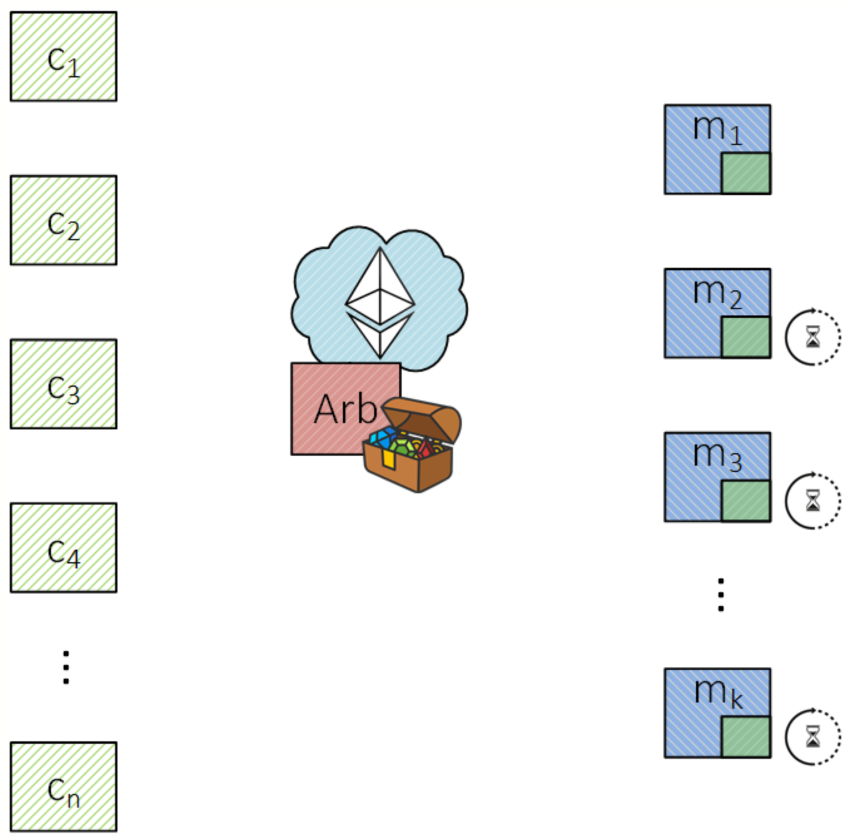
Snappy: Statekeeping Merchants



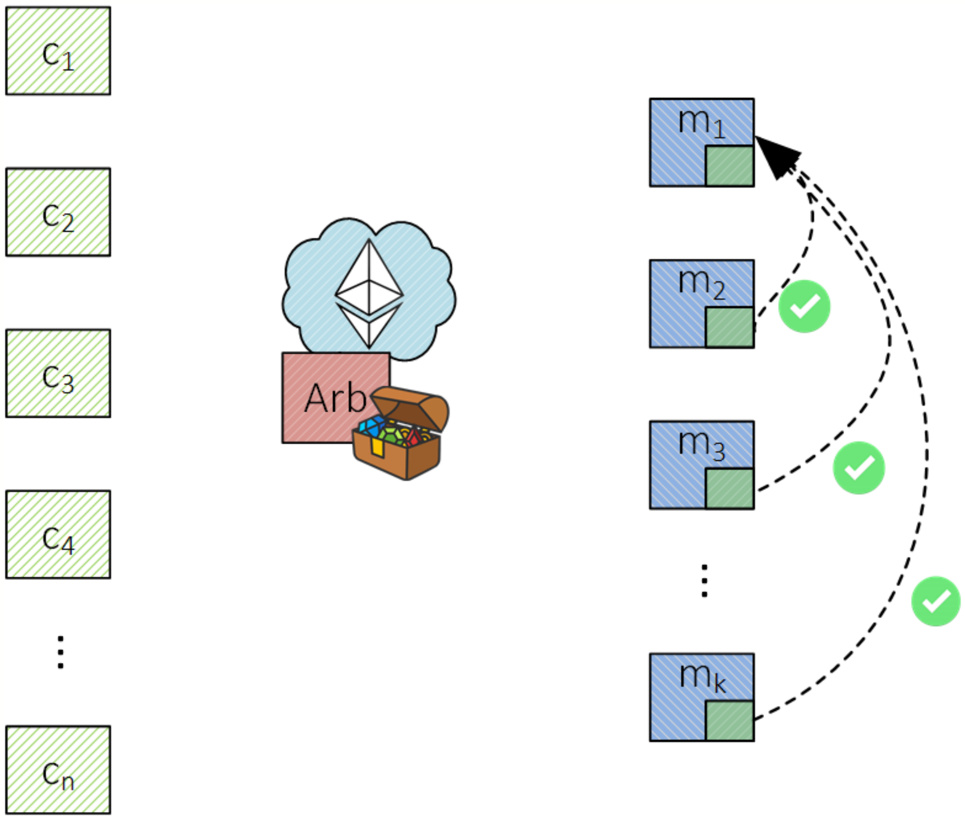
Snappy: Statekeeping Merchants



Snappy: Statekeeping Merchants

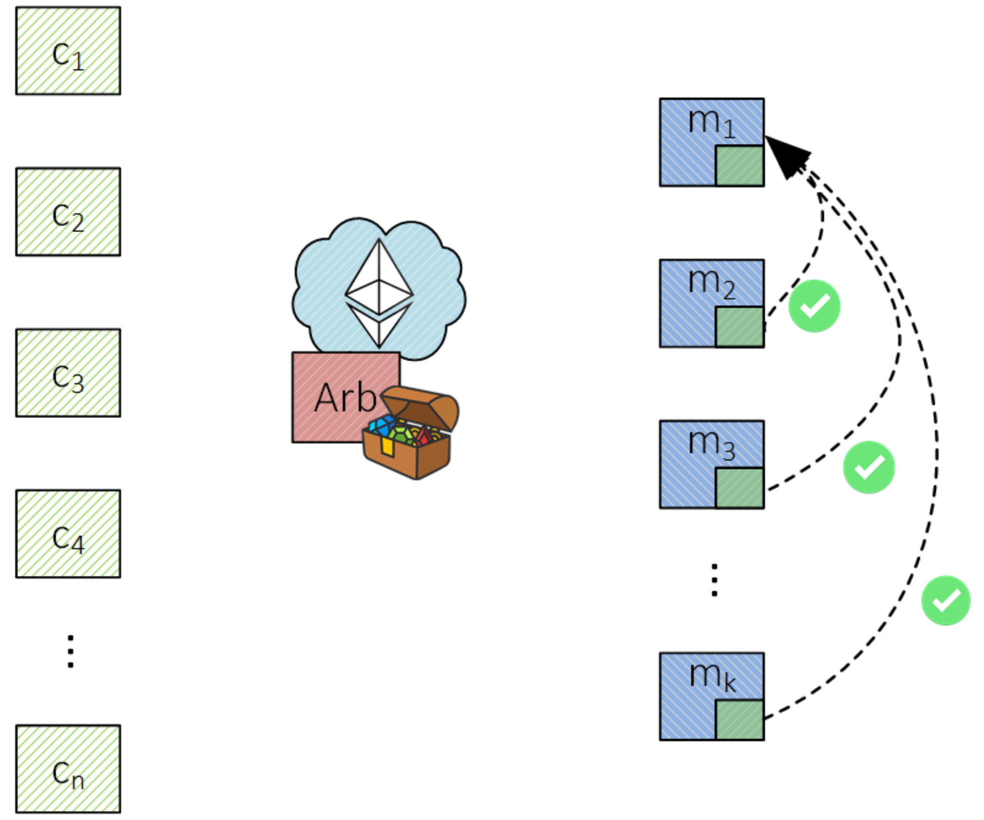


Snappy: Statekeeping Merchants



50%+1 Approvals

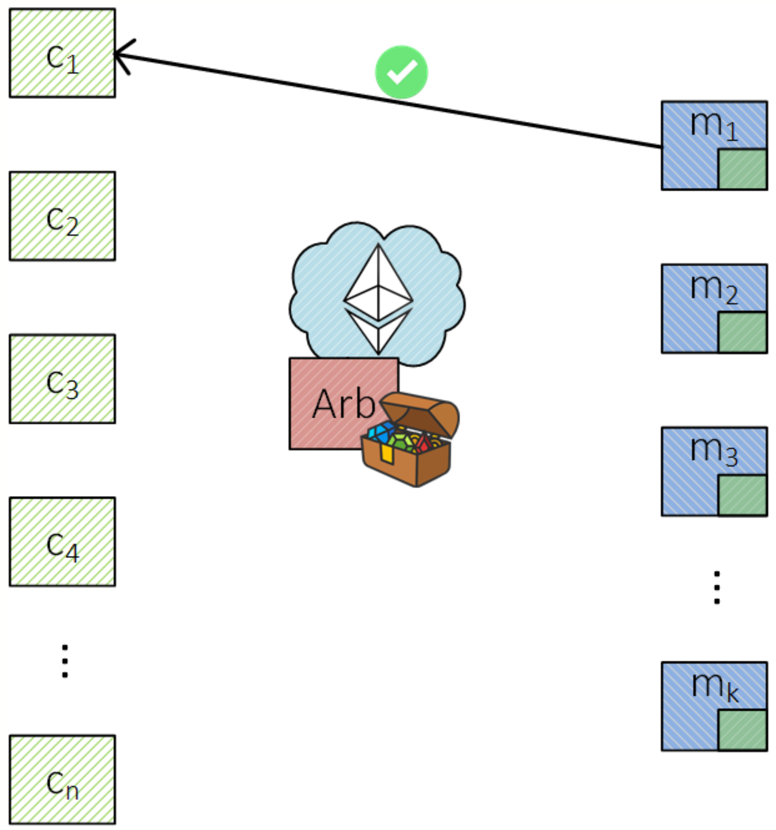
Snappy: Statekeeping Merchants



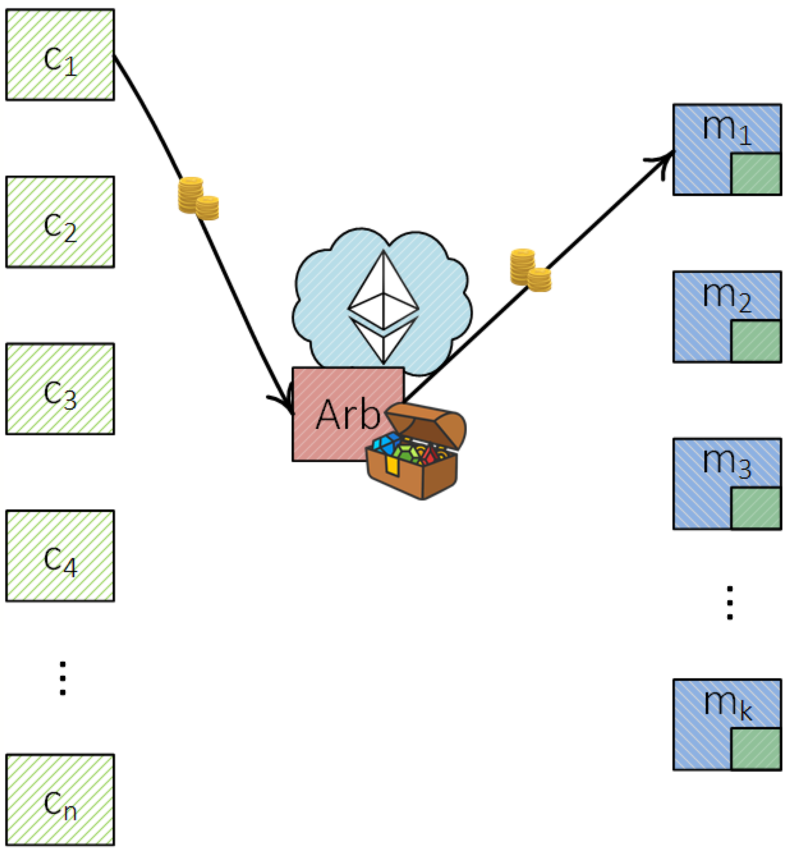
50%+1 Approvals

Approval: “I haven’t approved another transaction from c_1 with the same index number.”

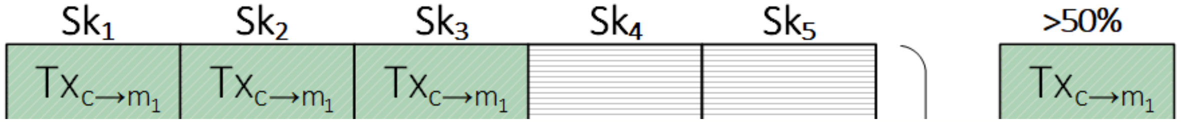
Snappy: Statekeeping Merchants



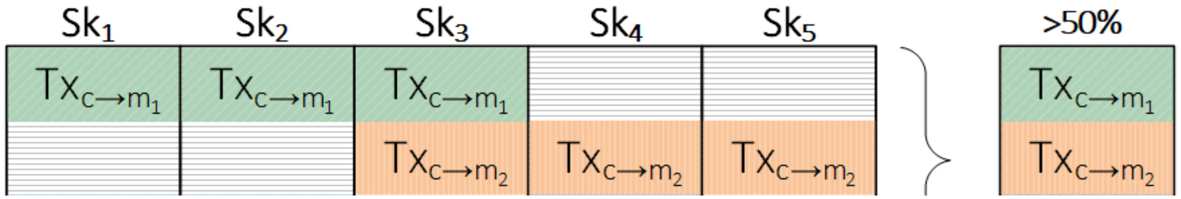
Snappy: Statekeeping Merchants



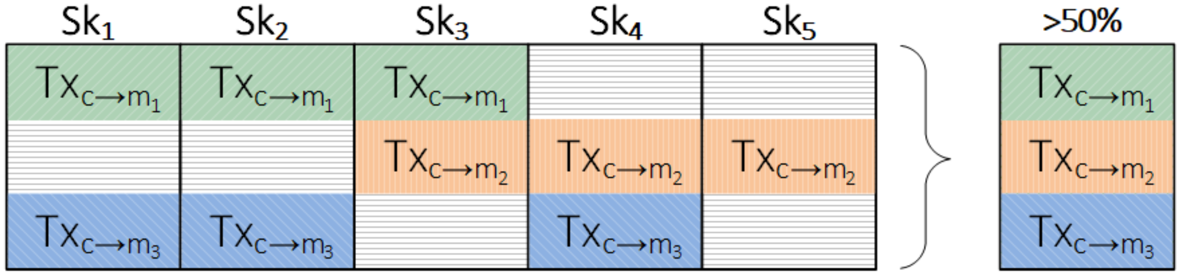
Proof of Merchant Equivocation



Proof of Merchant Equivocation

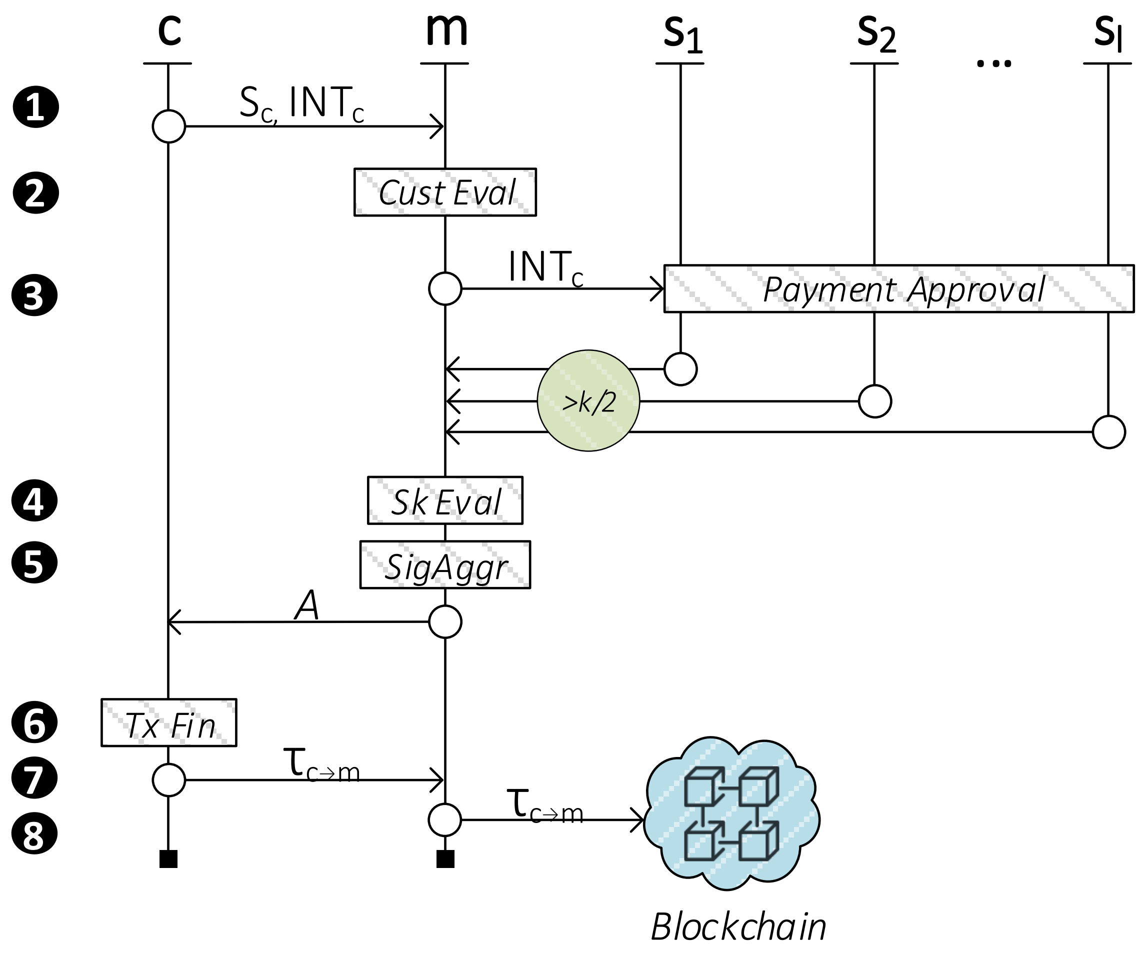


Proof of Merchant Equivocation



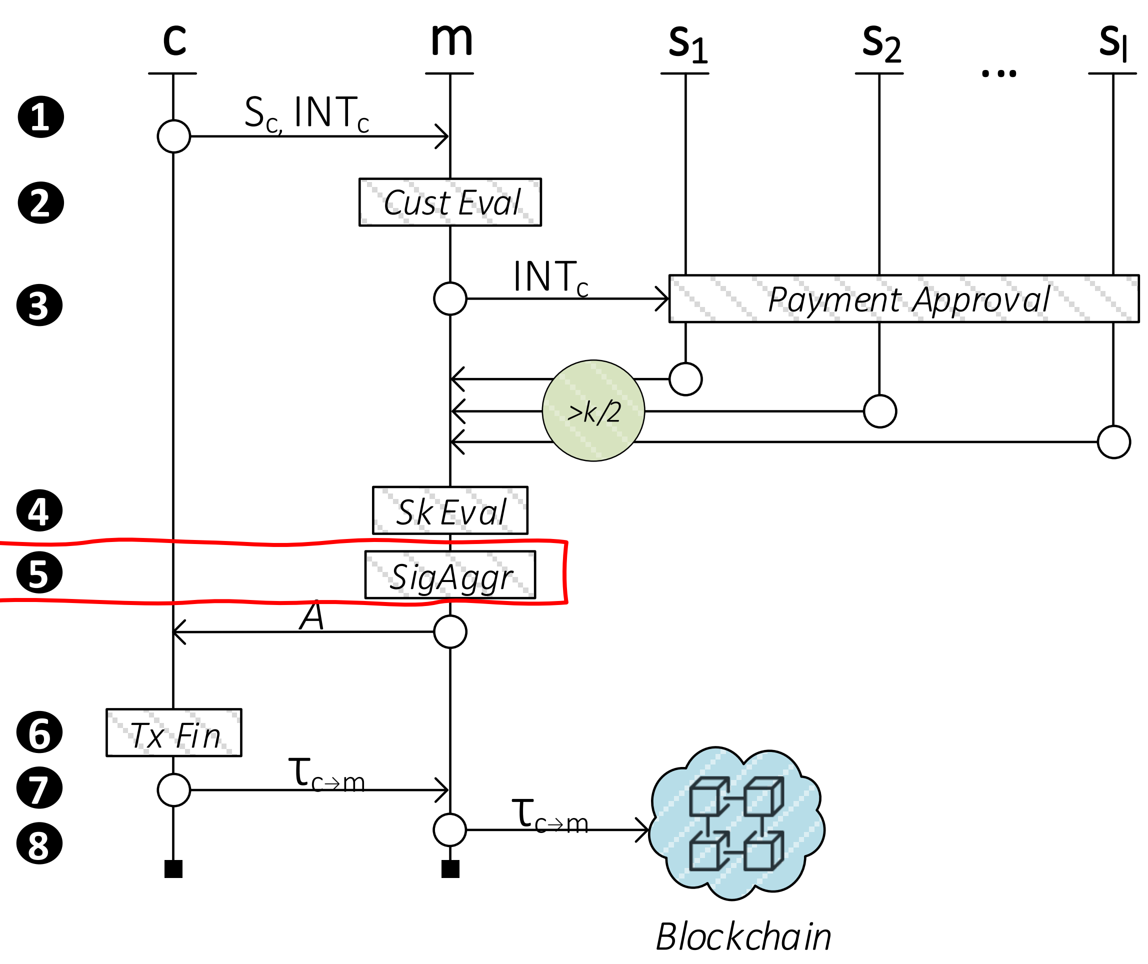
Approval Protocol

1. The customer initializes the payment.
2. Merchant verifies the collateral suffices.
3. Payment approval (50%+1).
4. Statekeeper evaluation.
5. Signature aggregation (e.g., BLS).
6. Customer signs final transaction.
7. Merchant verifies and completes checkout.
8. Transaction logged in blockchain and by the smartcontract.

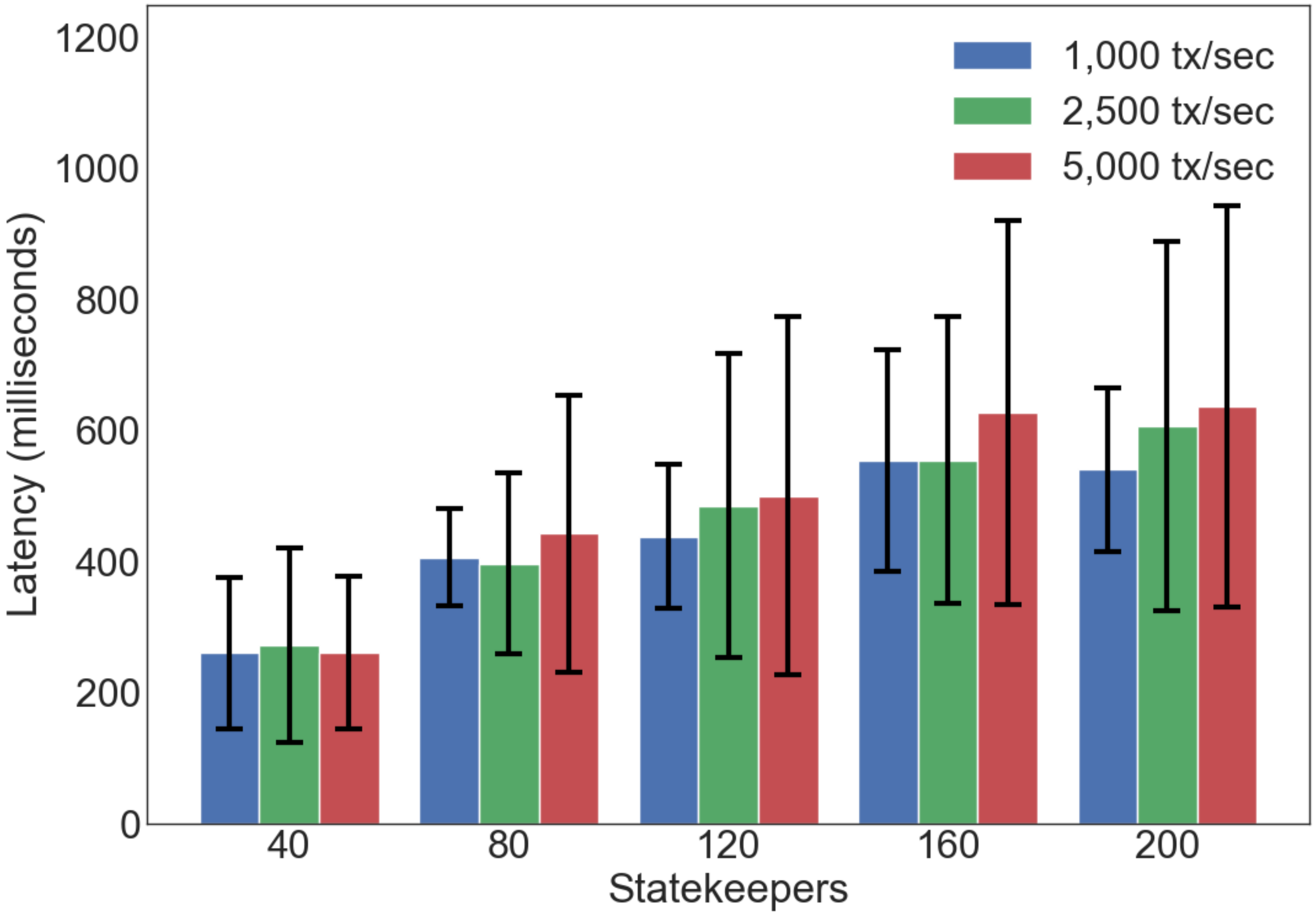


Approval Protocol

1. The customer initializes the payment.
2. Merchant verifies the collateral suffices.
3. Payment approval (50%+1).
4. Statekeeper evaluation.
5. Signature aggregation (e.g., BLS).
6. Customer signs final transaction.
7. Merchant verifies and completes checkout.
8. Transaction logged in blockchain and by the smartcontract.



Scalability: Latency



Scalability: Small Collaterals

- Only need to cover the expenditure within the latency period.
- Reusable.
- Flexible.
- Independent of the number of customers.

Takeaways

- ❖ An honest merchant never loses funds.
- ❖ Deployable on top of existing blockchains (e.g.,Ethereum).
- ❖ No additional trust assumptions.
- ❖ Small amount of locked in funds.
- ❖ Very low latency.

Thank you! Questions?

Snappy

Fast On-chain Payments with Practical Collaterals

Vasilios Mavroudis, Karl Wüst, Aritra Dhar, Kari Kostiainen, Srdjan Capkun