Fair contract signing: From protocols to sequences and graphs

Sjouke Mauw
University of Luxembourg
sjouke.mauw@uni.lu

(joint work with Barbara Kordy and Saša Radomirović)

February 5, 2013

Complete sequences

A sequence over a finite set $\mathcal{A}$ is complete if it contains all permutations of elements of $\mathcal{A}$ as a subsequence.

Example: complete sequence

The sequence D, B, D is complete over $\{B, D\}$.

Example: complete sequence

The sequence $D, B, D, \quad C, \quad D, B, D$ is complete over $\{B, C, D\}$.

Example: complete sequence

The sequence $D, B, D, \quad C, \quad D, B, D$ is complete over $\{B, C, D\}$.

D, B, D, C, D, B, D contains all permutations of $\{B, C, D\}$:

	D	B	D	C	D	B	D
DBC:	•	•		•			
DCB:	•			•		•	
BDC:		•	•	•			
BCD:		•		•	•		
CDB:				•	•	•	
CBD:				•		•	•

Constructing complete sequences

Let $\pi_1, \ldots, \pi_n$ be permutations of elements in $\mathcal{A}$. Then $\pi_1 \cdots \pi_n$ is complete. (Length n^2.)

Constructing complete sequences

Let $\pi_1, \ldots, \pi_n$ be permutations of elements in $\mathcal{A}$. Then $\pi_1 \cdots \pi_n$ is complete. (Length n^2.)

Choose permutations such that last element of π_i equals first element of π_{i+1}. Reduced sequence has length $n^2 - n + 1$.

Shortest complete sequences

Algorithms that produce complete sequences of size

$$n^2 - 2n + 4 \qquad (n \geq 3)$$

have been known since 1973.
This was the conjectured lower bound for complete sequences.
New lower bound:

$$\lceil n^2 - 7/3n + 19/3 \rceil \qquad (n \geq 7)$$

The problem has been open since 1972.

Multi-party contract signing (MPCS) protocols

- Purpose: Exchange signatures on a pre-agreed contract.

Multi-party contract signing (MPCS) protocols

- Purpose: Exchange signatures on a pre-agreed contract.
- Requirements: Fairness, Timeliness.

Multi-party contract signing (MPCS) protocols

- Purpose: Exchange signatures on a pre-agreed contract.
- Requirements: Fairness, Timeliness.

- Fairness:
  All honest signers get all signatures on the contract or nobody gets honest signers' signatures.

Multi-party contract signing (MPCS) protocols

- Purpose: Exchange signatures on a pre-agreed contract.
- Requirements: Fairness, Timeliness.

- Fairness:
  All honest signers get all signatures on the contract or nobody gets honest signers' signatures.
- Timeliness:
  No signer can be forced to wait endlessly.

Motivation

Alice plans a world trip on a budget.

Motivation

Alice plans a world trip on a budget.

- Needs to book hotels, transportation, event tickets.
- Independent companies.
- Cheap, non-refundable offers. Limited availability.

Motivation

Alice plans a world trip on a budget.

- Needs to book hotels, transportation, event tickets.
- Independent companies.
- Cheap, non-refundable offers. Limited availability.

Once offers are collected, either all bookings should be made or none of them.

Motivation

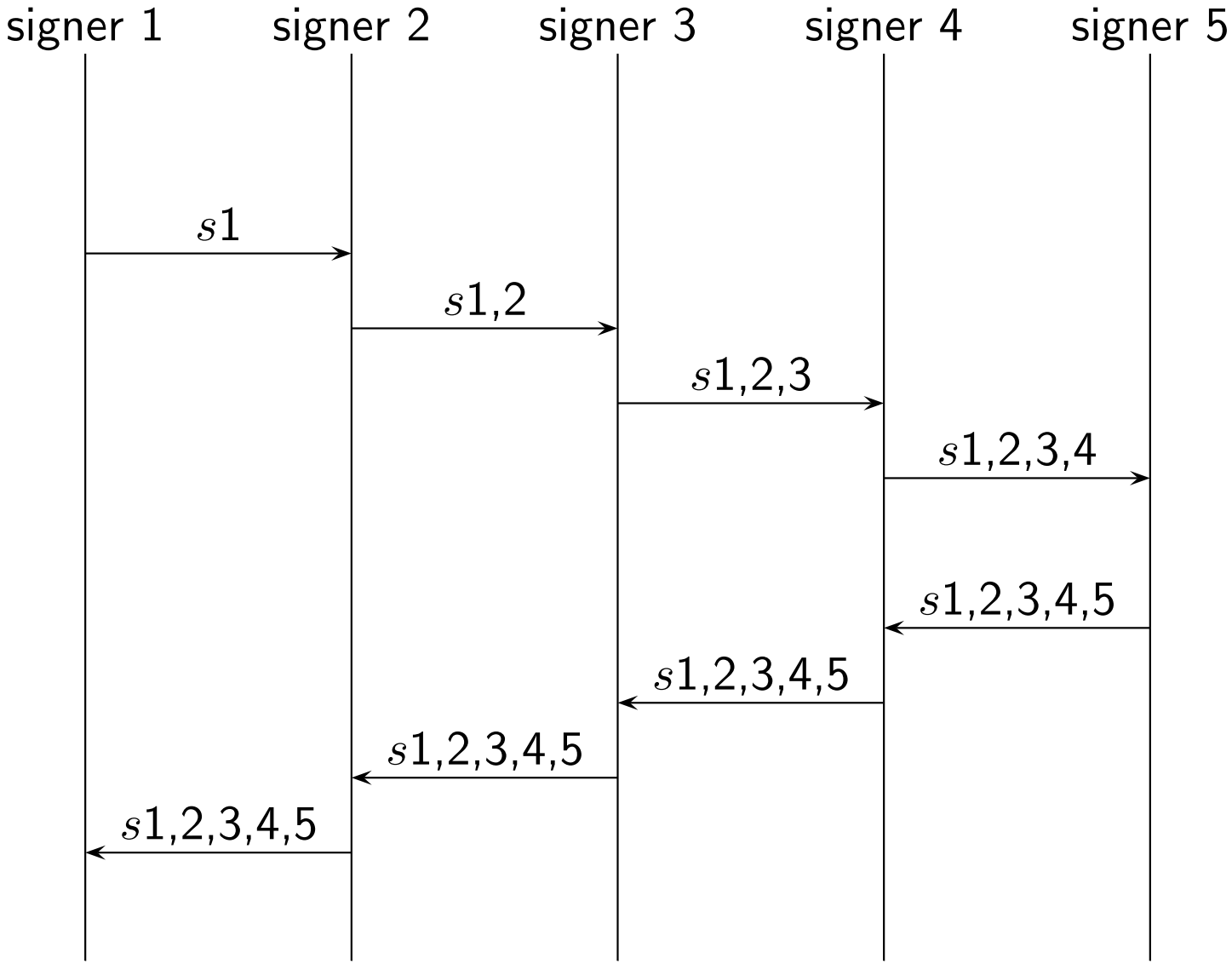
Alice plans a world trip on a budget.

- Needs to book hotels, transportation, event tickets.
- Independent companies.
- Cheap, non-refundable offers. Limited availability.

Once offers are collected, either all bookings should be made or none of them.

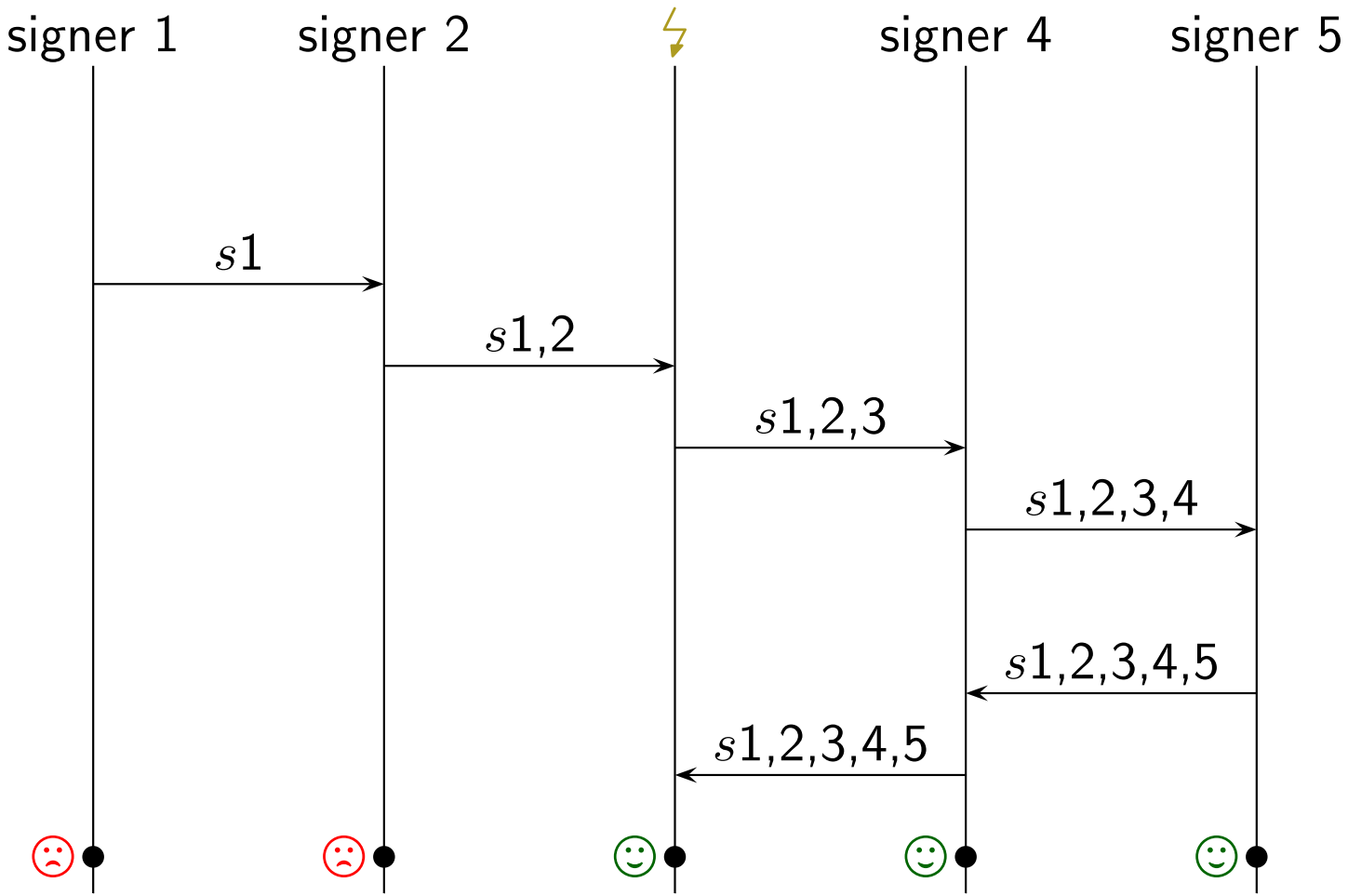
Solution: Alice sets up one single contract with all companies she would like to buy a service from.

A simple but unfair protocol



s1,2,3 = contract signed by signers 1, 2 and 3.

Why this protocol is not fair



Signer 3 is dishonest and obtains a fully signed contract, while signers 1 and 2 have no (fully) signed contract.

Alice's steps

To obtain a fair contract signing protocol, Alice needs

1. to construct a complete sequence,
2. to transform the sequence to a protocol,
3. access to a trusted third party (TTP) in case a failure occurs,
4. a reason why the above ensures fairness.

Why the TTP?

Assumptions:

- Asynchronous communication.
- Deterministic protocol.

$\Rightarrow$ TTP is necessary in order to achieve fairness.
(Even & Yacobi, 1980.)

Optimistic protocols

Optimistic protocols make use of TTP only in case of a failure.

Optimistic protocols

Optimistic protocols make use of TTP only in case of a failure.
Consist of two sub-protocols:

- Main protocol: Exchange of *promises* and *signatures* between signers.

Optimistic protocols

Optimistic protocols make use of TTP only in case of a failure.
Consist of two sub-protocols:

- Main protocol: Exchange of *promises* and *signatures* between signers.

- Resolve protocol: Signer sends all received messages to TTP if a failure occurs.

  TTP decides

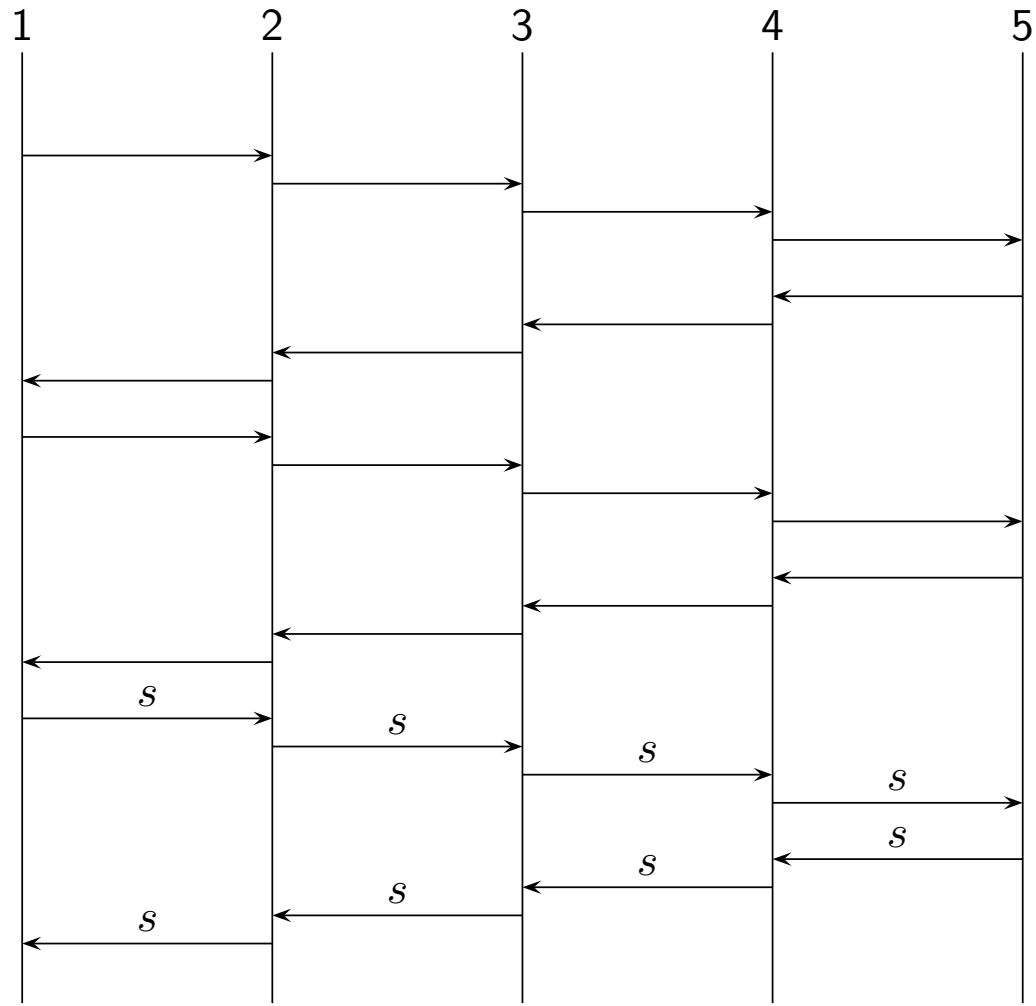
  - Abort: TTP sends meaningless abort token to requesting signer.
  - Resolve: TTP converts promises to signed contract and sends it to requesting signer.

Approach

Introduce one or more rounds of promises before the final signatures are placed.

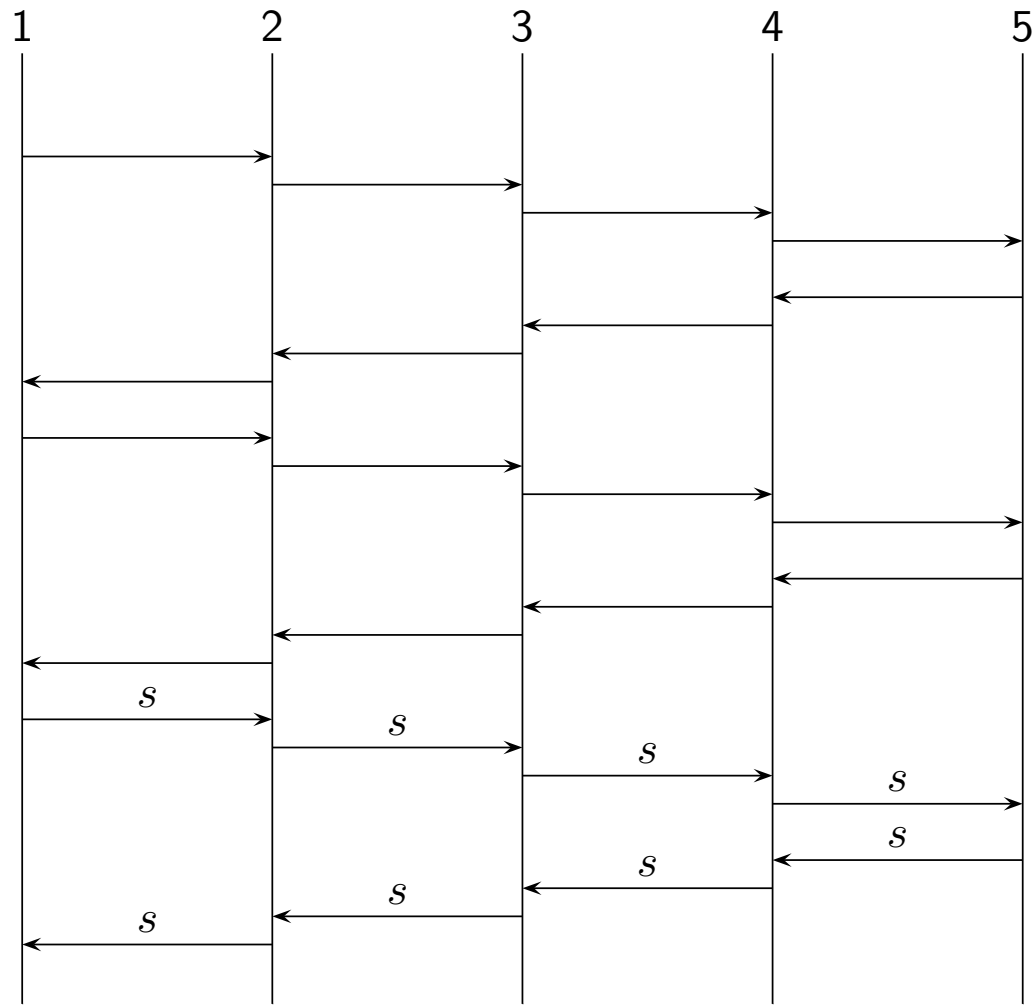
If you have someone's promise, but you don't receive his signature, you send the promise to the TTP who converts it into a signed contract.

An MPCS main protocol (unfair)



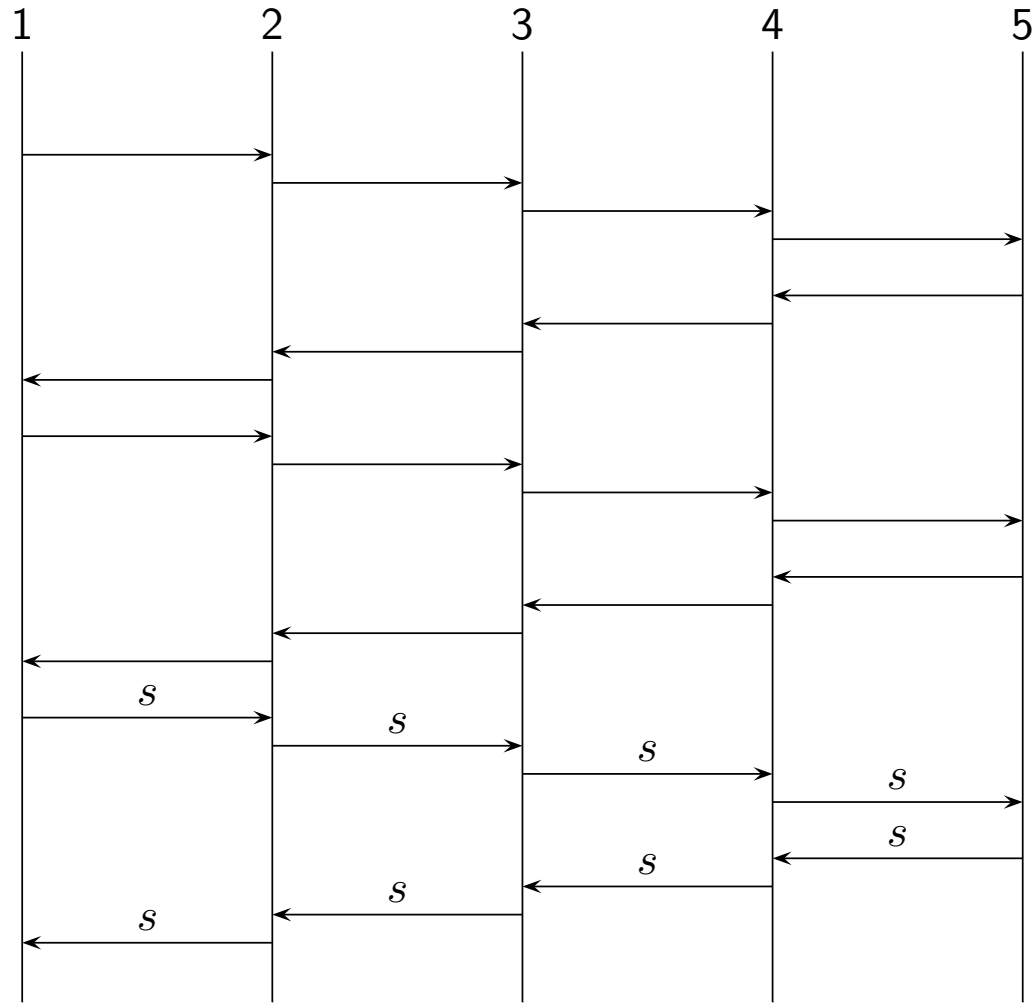
2 rounds of promises, 1 round of signatures.

Related sequence



1234 5432 1234 5432 1234 54321

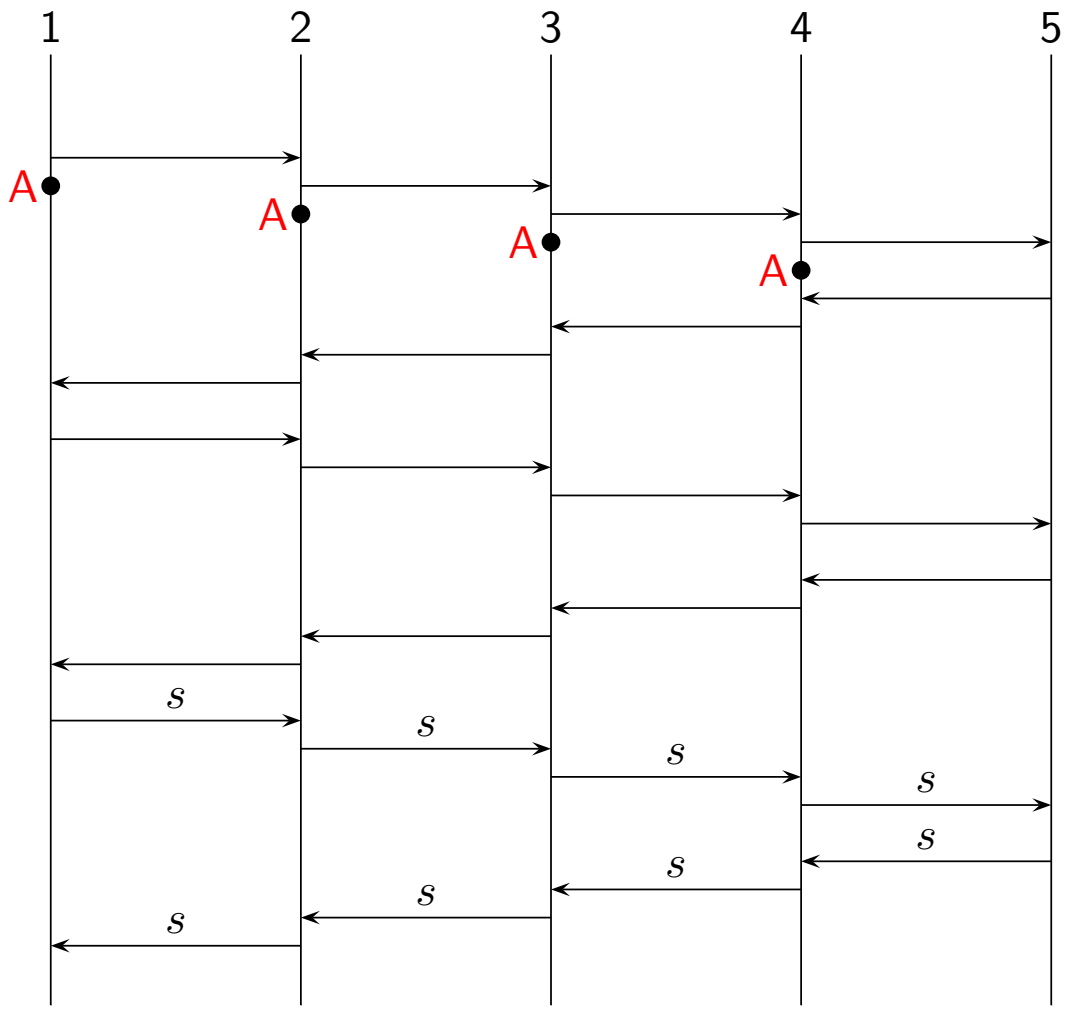
Related sequence



(1234) **5432 1234 5432 1234 5** (4321)

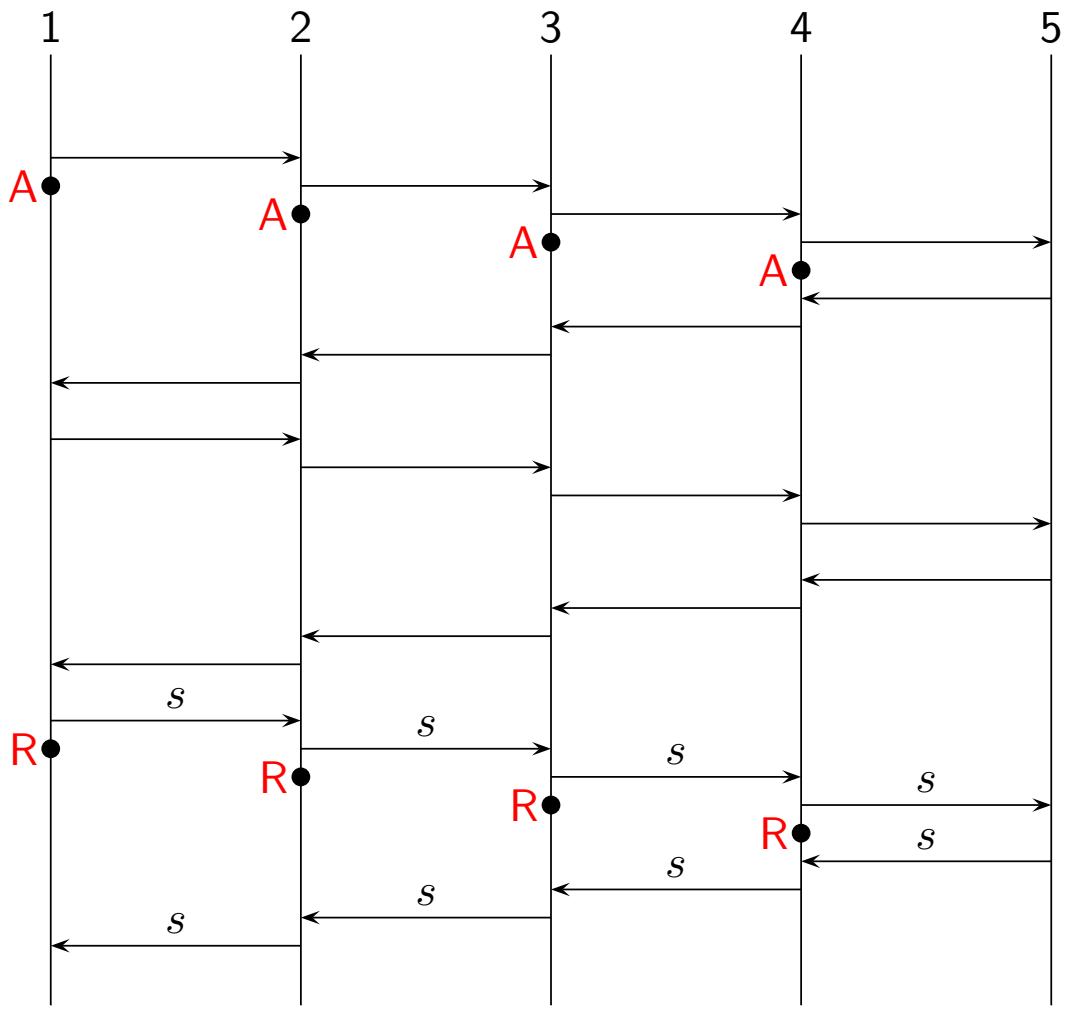
Bolded part lacks subsequence 45231

TTP Resolve decisions: forced abort



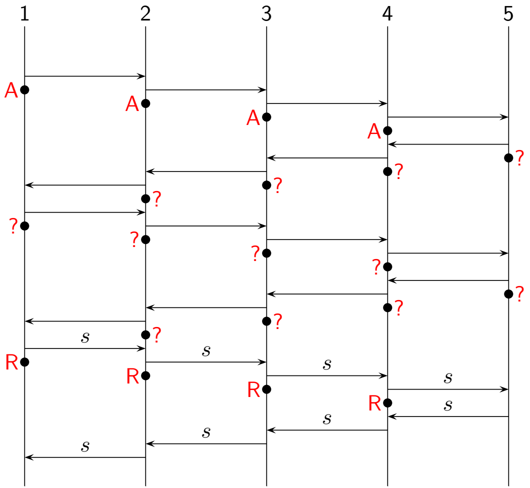
A = If called from here, TTP is forced to abort

TTP Resolve decisions: forced resolve



R = If called from here, TTP is forced to resolve

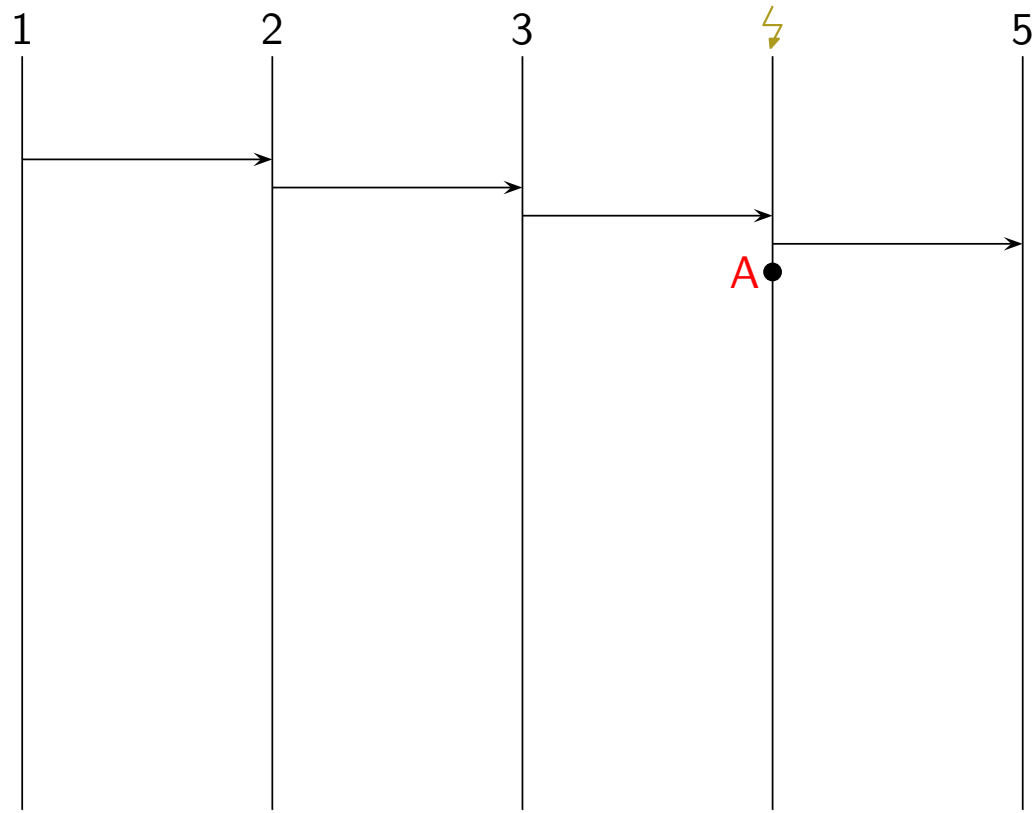
TTP Resolve decisions: rest depends on history



? = If already aborted for honest signer, then abort. Otherwise resolve.

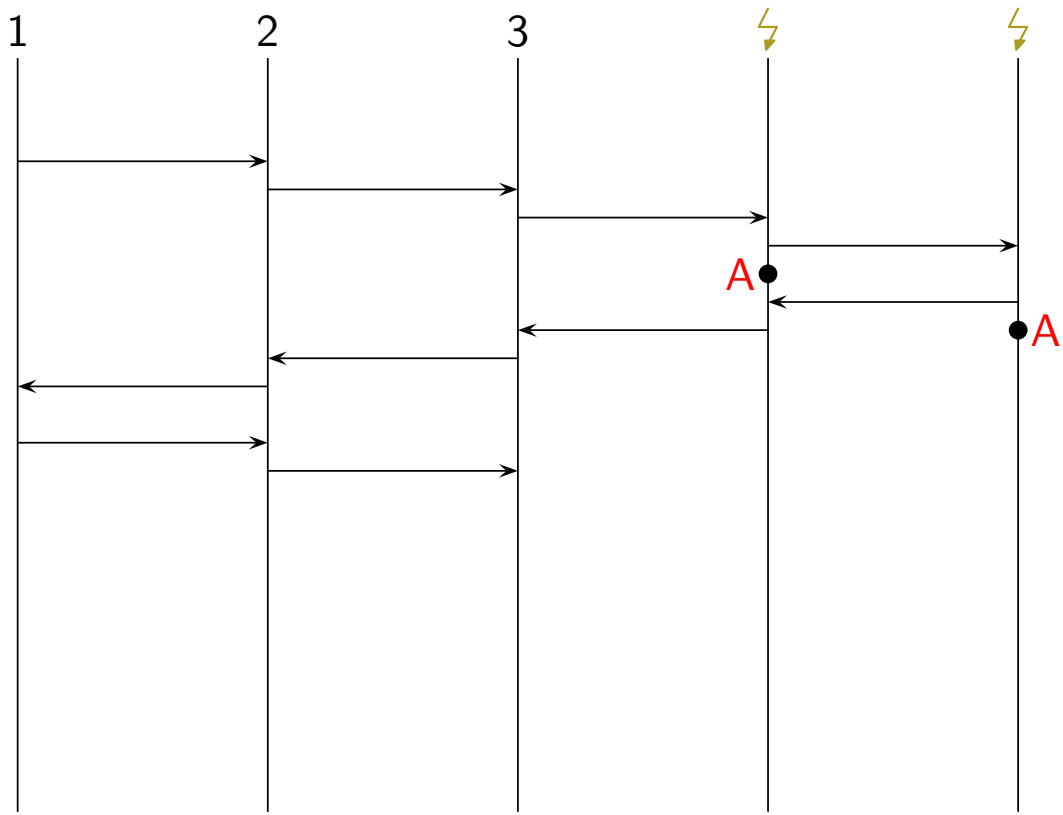
Abort chaining (**4**5231)

How a sequence of (dishonest) abort requests can fool a signer.



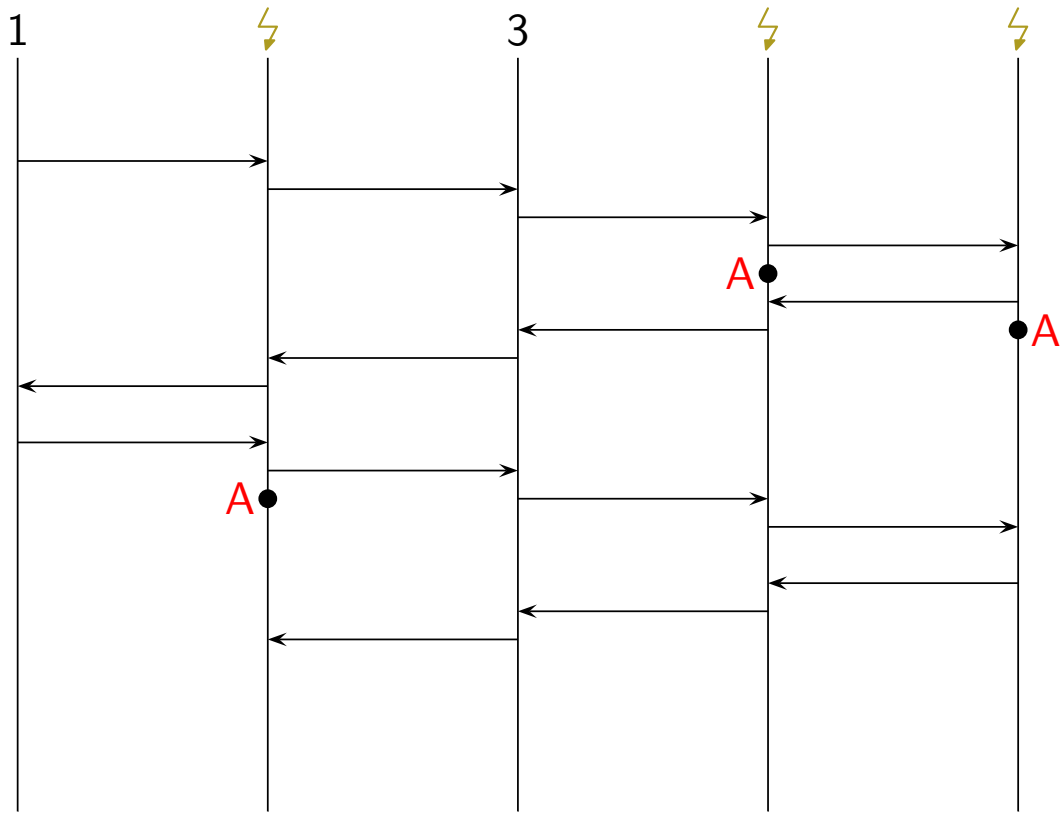
Dishonest signer 4 sends abort request and receives abort reply.
Dishonesty = request abort/resolve, but still continue the main protocol.

Abort chaining (4**5**231)



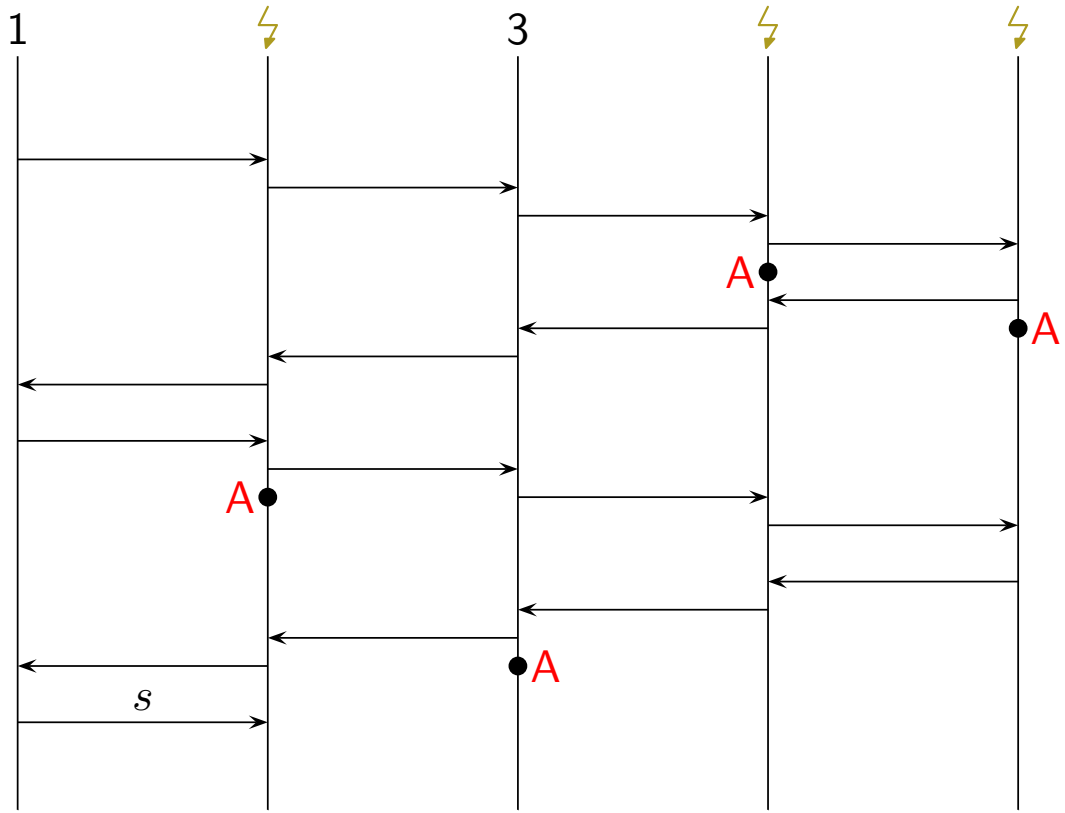
Dishonest signer 5 sends abort request. TTP could send resolved contract to 5, but because he has previously sent an abort to 4 and because he doesn't know that 4 is dishonest, he sends an abort to 5.

Abort chaining (45231)



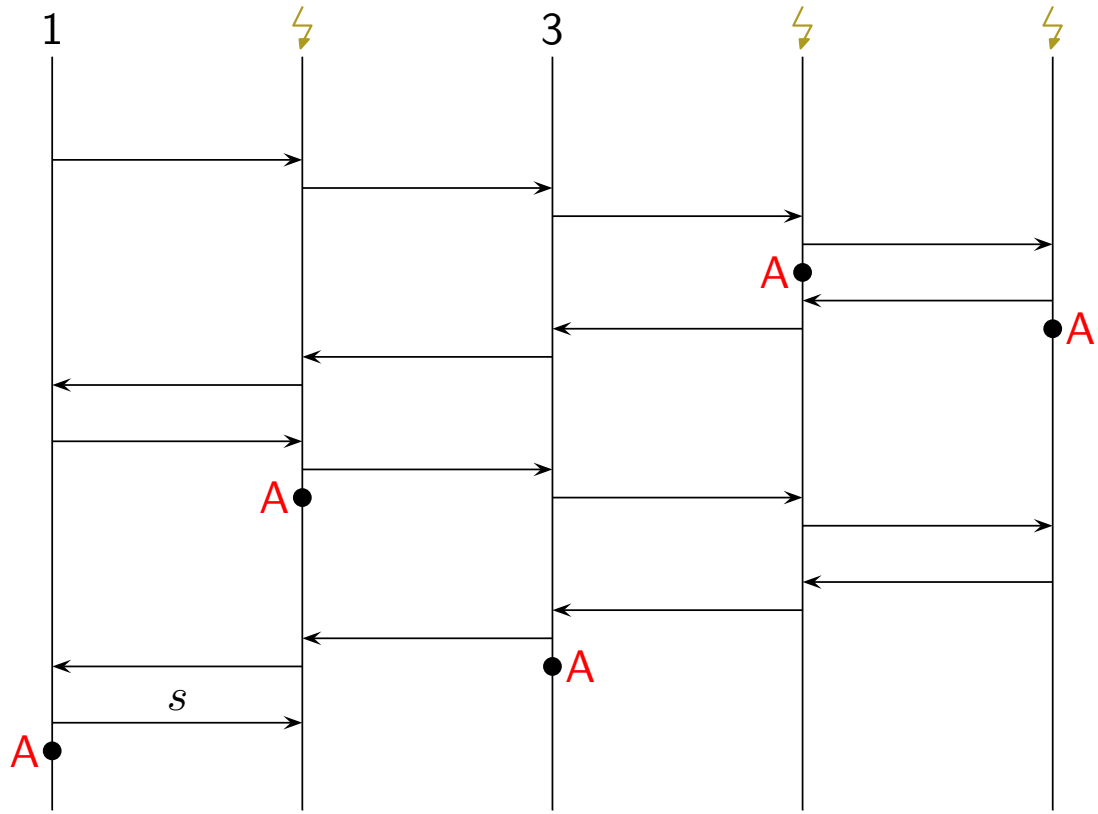
Dishonest signer 2 sends abort request and receives abort reply.
TTP discovers that signer 4 is dishonest.
TTP doesn't know (yet) that signer 5 is dishonest.

Abort chaining (45231)



Dishonest signer 2 does not send his signature.
Honest signer 3 does not receive a signature and sends a request to the TTP who answers with abort.
TTP discovers that signer 5 is dishonest.
TTP doesn't know (yet) that signer 2 is dishonest.

Abort chaining (45231)



Now TTP knows that signer 2 is dishonest.
TTP doesn't know if signer 3 is honest or not.
Signer 1 requests resolve and TTP must reply abort.
But signer 1 already sent his signature → UNFAIR

Solution

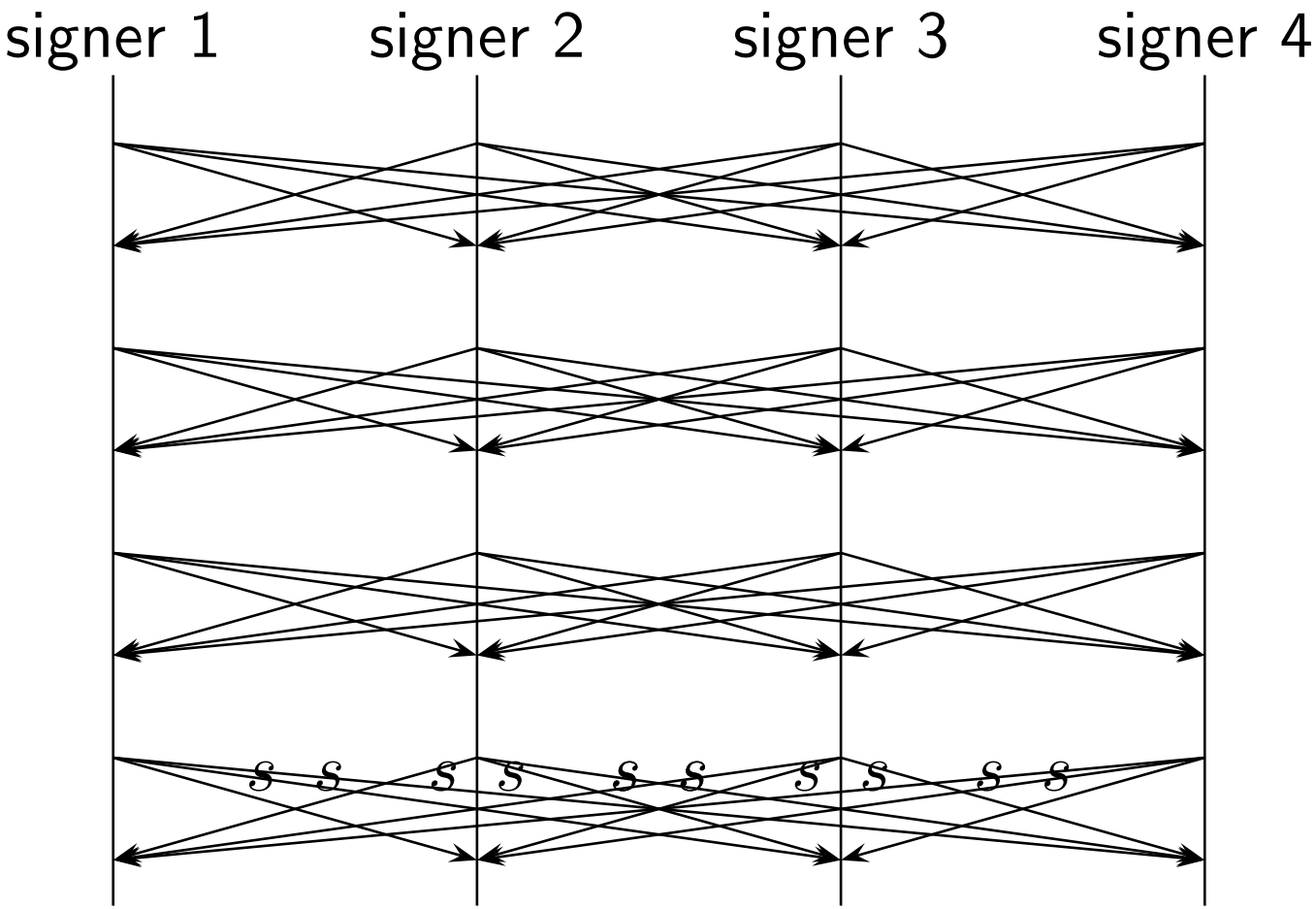
One more round of promises prevents an abort chaining attack.
In this case 3 rounds of promises suffices.

This corresponds to a complete sequence.
(1234) **5432 1234 5432 1234 5432 1234 5** (4321)

Shortest complete sequences provide novel protocols with fewer messages than those published in literature.

Generalization

Until now only *linear* protocols. What if we allow parallelism?



Directed Acyclic Graphs

Fair parallel protocols relate to complete DAGs.

Complete DAGs generalise complete sequences.

Conclusion

- Every complete sequence gives rise to a fair optimistic MPCS protocol.

Conclusion

- Every complete sequence gives rise to a fair optimistic MPCS protocol.
- Abundance of complete sequences.
  $\Rightarrow$ abundance of fair optimistic MPCS protocols.

Conclusion

- Every complete sequence gives rise to a fair optimistic MPCS protocol.
- Abundance of complete sequences.
  $\Rightarrow$ abundance of fair optimistic MPCS protocols.
- Shortest complete sequences give protocols with optimal *message* complexity.

Conclusion

- Every complete sequence gives rise to a fair optimistic MPCS protocol.
- Abundance of complete sequences.
  $\Rightarrow$ abundance of fair optimistic MPCS protocols.
- Shortest complete sequences give protocols with optimal *message* complexity.
- Complete DAGs generalize complete sequences, relate to parallel protocols.

Conclusion

- Every complete sequence gives rise to a fair optimistic MPCS protocol.
- Abundance of complete sequences.
  $\Rightarrow$ abundance of fair optimistic MPCS protocols.
- Shortest complete sequences give protocols with optimal *message* complexity.
- Complete DAGs generalize complete sequences, relate to parallel protocols.
- Minimal complete DAGs give protocols with optimal *time* complexity.

Conclusion

- Every complete sequence gives rise to a fair optimistic MPCS protocol.
- Abundance of complete sequences.
  $\Rightarrow$ abundance of fair optimistic MPCS protocols.
- Shortest complete sequences give protocols with optimal *message* complexity.
- Complete DAGs generalize complete sequences, relate to parallel protocols.
- Minimal complete DAGs give protocols with optimal *time* complexity.
- New research question: what are minimal complete DAGs and how to generate them?

Conclusion

- Every complete sequence gives rise to a fair optimistic MPCS protocol.
- Abundance of complete sequences.
  $\Rightarrow$ abundance of fair optimistic MPCS protocols.
- Shortest complete sequences give protocols with optimal *message* complexity.
- Complete DAGs generalize complete sequences, relate to parallel protocols.
- Minimal complete DAGs give protocols with optimal *time* complexity.
- New research question: what are minimal complete DAGs and how to generate them?
- Still open: What are the shortest complete sequences over a finite set?

Thanks for your attention.

Questions?