

## Contract-Signing Protocols

John Mitchell  
Stanford

## Contract Signing

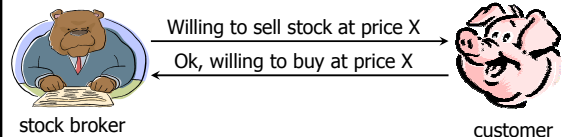
- ◆ Two parties want to sign a contract
  - Multi-party signing is more complicated
- ◆ The contract is known to both parties
  - The protocols we will look at are *not* for contract negotiation (e.g., auctions)
- ◆ The attacker could be
  - Another party on the network
  - The "person" you think you want to sign a contract with

## Example



- ◆ Both parties want to sign the contract
- ◆ Neither wants to commit first

## Another example: stock trading



- ◆ Why signed contract?
  - Suppose market price changes
  - Buyer or seller may want proof of agreement

## Network is Asynchronous

- ◆ Physical solution
  - Two parties sit at table
  - Write their signatures simultaneously
  - Exchange copies
- ◆ Problem
  - How to sign a contract on a network?

Fair exchange: general problem of exchanging information so both succeed or both fail

## Fundamental limitation

- ◆ Impossibility of consensus
  - *Very weak consensus is not solvable if one or more processes can be faulty*
- ◆ Asynchronous setting
  - Process has *initial* 0 or 1, and eventually *decides* 0 or 1
  - *Weak termination*: some correct process decides
  - *Agreement*: no two processes decide on different values
  - *Very weak validity*: there is a run in which the decision is 0 and a run in which the decision is 1
- ◆ Reference
  - M. J. Fischer, N. A. Lynch and M. S. Paterson, *Impossibility of Distributed Consensus with One Faulty Process*, J ACM 32(2):374-382 (April 1985).

## FLP Partial Intuition

### ◆Quote from paper:

- The asynchronous commit protocols in current use all seem to have a "window of vulnerability"- an interval of time during the execution of the algorithm in which the delay or inaccessibility of a single process can cause the entire algorithm to wait indefinitely. It follows from our impossibility result that every commit protocol has such a "window," confirming a widely believed tenet in the folklore.

## Implication for fair exchange

### ◆Need a trusted third party (TTP)

- It is impossible to solve strong fair exchange without a trusted third party. The proof is by relating strong fair exchange to the problem of consensus and adapting the impossibility result of Fischer, Lynch and Paterson.

### ◆Reference

- H. Pagnia and F. C. Gärtner, On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Darmstadt University of Technology, March 1999

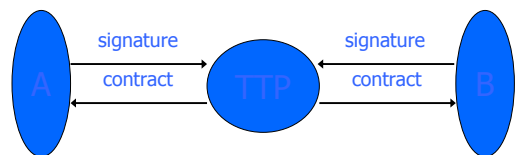
## Two forms of contract signing

### ◆Gradual-release protocols

- Alice and Bob sign contract
- Exchange signatures a few bits at a time
- Issues
  - Signatures are verifiable
  - Work required to guess remaining signature decreases
  - Alice, Bob must be able to verify that what they have received so far is part of a valid signature

### ◆Add trusted third party

## Easy TTP contract signing



### ◆Problem

- TTP is bottleneck
- Can we do better?

## Optimistic contract signing

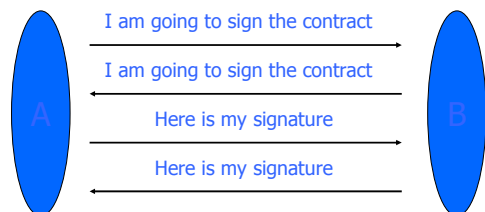
### ◆Use TTP only if needed

- Can complete contract signing without TTP
- TTP will make decisions if asked

### ◆Goals

- Fair: no one can cheat the other
- Timely: no one has to wait indefinitely (assuming that TTP is available)
- Other properties ...

## General protocol outline



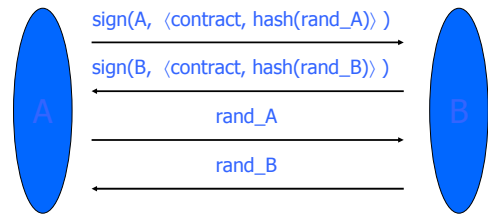
### ◆Trusted third party can force contract

- Third party can declare contract binding if presented with first two messages.

## Commitment (idea from crypto)

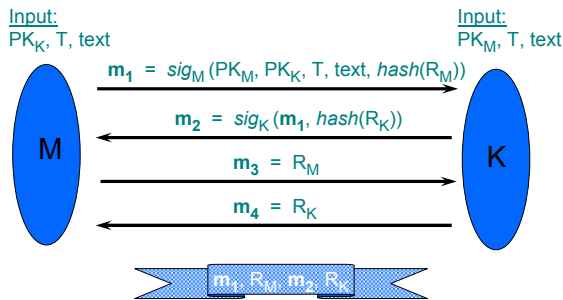
- ◆ Cryptographic hash function
  - Easy to compute function  $f$
  - Given  $f(x)$ , hard to find  $y$  with  $f(y)=f(x)$
  - Hard to find pairs  $x, y$  with  $f(y)=f(x)$
- ◆ Commit
  - Send  $f(x)$  for randomly chosen  $x$
- ◆ Complete
  - Reveal  $x$

## Refined protocol outline



- ◆ Trusted third party can force contract
  - Third party can declare contract binding by signing first two messages.

## Optimistic Protocol [Asokan, Shoup, Waidner]



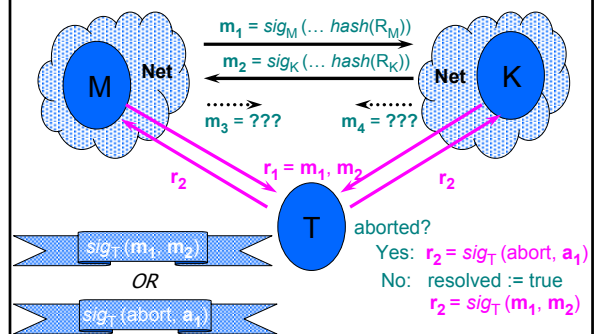
## Asokan-Shoup-Waidner Outcomes

- ◆ Contract from normal execution
  - $m_1, R_M, m_2, R_K$
- ◆ Contract issued by third party
  - $sig_T(m_1, m_2)$
- ◆ Abort token issued by third party
  - $sig_T(\text{abort}, a_1)$

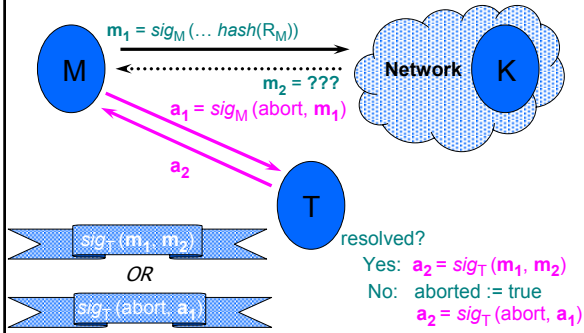
## Role of Trusted Third Party

- ◆ T can issue a replacement contract
  - Proof that both parties are committed
- ◆ T can issue an abort token
  - Proof that T will not issue contract
- ◆ T acts only when requested
  - decides whether to abort or resolve on the first-come-first-serve basis
  - only gets involved if requested by M or K

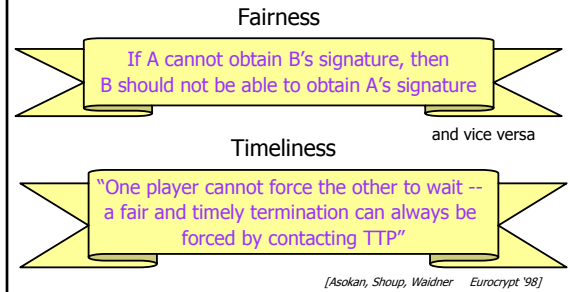
## Resolve Subprotocol



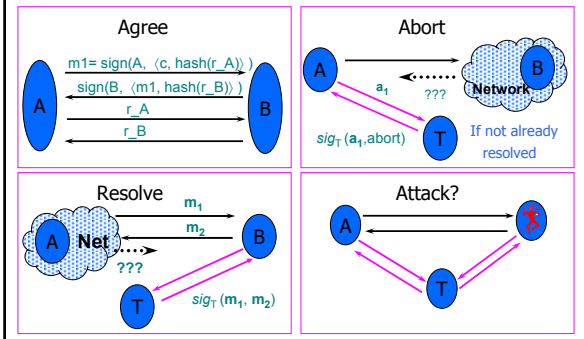
## Abort Subprotocol



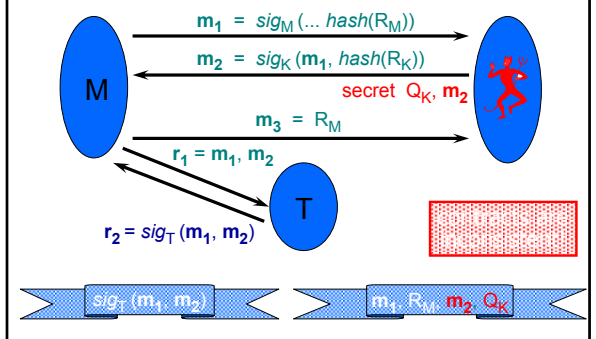
## Fairness and Timeliness



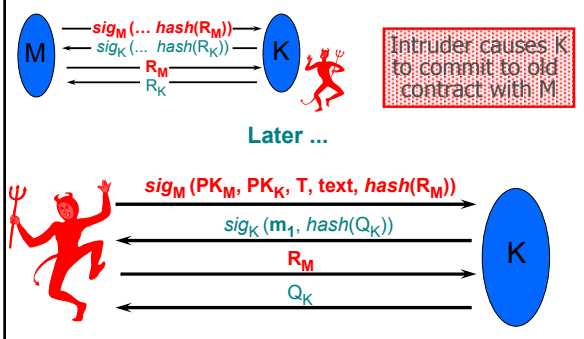
## Asokan-Shoup-Waidner protocol



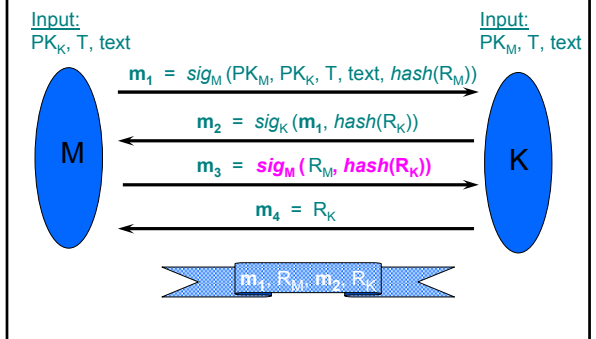
## Attack



## Replay Attack



## Fixing the Protocol

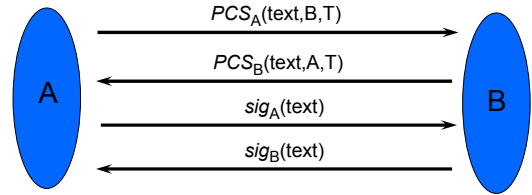


## Desirable properties

- ◆ Fair
  - If one can get contract, so can other
- ◆ Accountability
  - If someone cheats, message trace shows who cheated
- ◆ Abuse free
  - No party can show that they can determine outcome of the protocol

[Garay, Jakobsson, MacKenzie]

## Abuse-Free Contract Signing

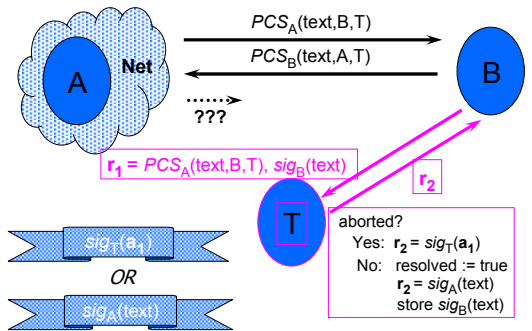


- ◆ Private Contract Signature
  - Special cryptographic primitive
  - B cannot take msg from A and show to C
  - T converts signatures, does not use own

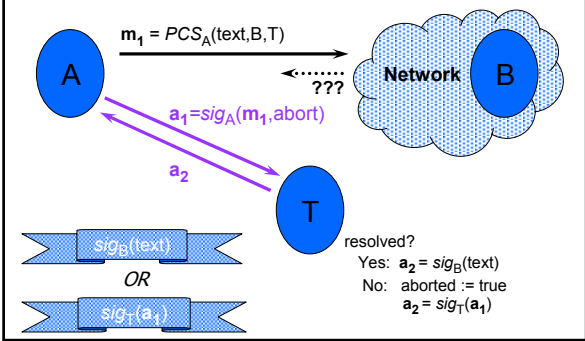
## Role of Trusted Third Party

- ◆ T can convert PCS to regular signature
  - Resolve the protocol if necessary
- ◆ T can issue an abort token
  - Promise not to resolve protocol in future
- ◆ T acts only when requested
  - decides whether to abort or resolve on a first-come-first-served basis
  - only gets involved if requested by A or B

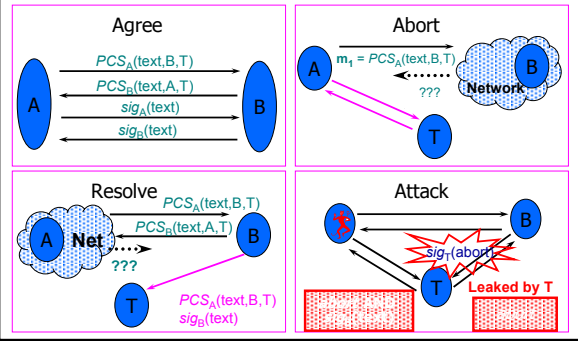
## Resolve Subprotocol



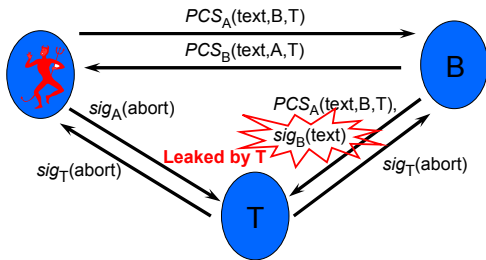
## Abort Subprotocol



## Garay, Jakobsson, MacKenzie

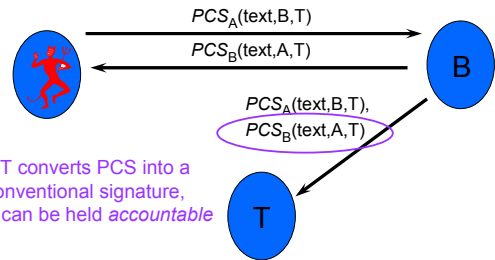


## Attack



abort AND  $sig_B(\text{text})$

## Repairing the Protocol



If T converts PCS into a conventional signature, T can be held accountable

## Balance

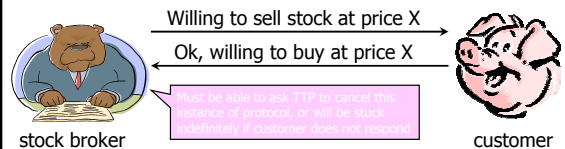
No party should be able to **unilaterally** determine the outcome of the protocol

Balance may be violated even if basic fairness is satisfied!

Stock sale example: there is a point in the protocol where the broker can **unilaterally** choose whether the sale happens or not

Can a timely, optimistic protocol be fair AND balanced?

## Advantage



Can go ahead and complete the sale, OR can still ask TTP to cancel (TTP doesn't know customer has responded)

Optimistically waits for broker to respond ...

Chooses whether deal will happen: does not have to commit stock for sale, can cancel if sale looks unprofitable

Cannot back out of the deal: must commit money for stock

## "Abuse free": as good as it gets

### ◆ Specifically:

- One signer always has an advantage over the other, no matter what the protocol is
- Best case: signer with advantage cannot prove it has the advantage to an outside observer

## Theorem

- ◆ In any fair, optimistic, timely contract-signing protocol, if one player is optimistic\*, the other player has an advantage.

\* optimistic player: waits a little before going to the third party

## Abuse-Freeness

~~Balance~~ impossible ☹

No party should be able to unilaterally determine the outcome of the protocol

Abuse-Freeness

No party should be able to **prove** that it can unilaterally determine the outcome of the protocol

[Garay, Jakobsson, MacKenzie Crypto '99]

## How to prove something like this?

- ◆ Define "protocol"
  - Program for Alice, Bob, TTP
  - Each move depends on
    - Local State (what's happened so far)
    - Message from network
    - Timeout
- ◆ Consider possible optimistic runs
- ◆ Show someone gets advantage

## Key idea (omitting many subtleties)

- ◆ Define "power" of a signer (A or B) in a state  $s$

$$\text{Power}_A(s) = \begin{cases} 2 & \text{if A can get contract by reading a message already in network, doing internal computation} \\ 1 & \text{if A can get contract by communicating with TTP, assuming B does nothing} \\ 0 & \text{otherwise} \end{cases}$$

- ◆ Look at optimistic transition  $s \rightarrow s'$  where  $\text{Power}_B(s) = 1 > \text{Power}_B(s') = 0$ .

## Advantage (intuition for main argument)

- ◆ If  $\text{Power}_B(s) = 0 \rightarrow \text{Power}_B(s') = 1$  then
  - This is result of some move by A
    - $\text{Power}_B(s) = 0$  means B cannot get contract without B's help
  - The move by A is not a message to TTP
    - The proof is for an *optimistic* protocol, so we are thinking about a run without msg to T
  - B could abort in state  $s$ 
    - We assume protocol is timely and fair: B must be able to do something, cannot get contract
  - B can still abort in  $s'$ , so B has advantage!

## Conclusions

- ◆ Online contract signing is subtle
  - Fair
  - Abuse-free
  - Accountability
- ◆ Several interdependent subprotocols
  - Many cases and interleavings
- ◆ Finite-state tools great for case analysis!
  - Find bugs in protocols proved correct
- ◆ Proving properties of all protocols is harder
  - Understand what is possible and what is not