CS 259

Contract-Signing Protocols

J. Mitchell
V. Shmatikov

Next few lectures

- Today
  - Contract-signing protocols
- Thursday
  - More about contract signing, probability
- Next Tues
  - Probabilistic model checking
- Next Thurs
  - Homework due; think about projects

After this week, cover tools and protocol examples together

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

Immunity deal

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

Another example: stock trading

Willing to sell stock at price X

Ok, willing to buy at price X

stock broker

customer

- 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.

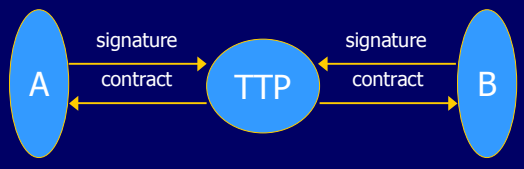
Implication for fair exchange

- 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

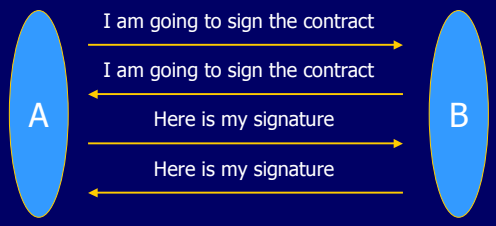


- Problem
  - TTP is bottleneck
  - Can we do better?

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

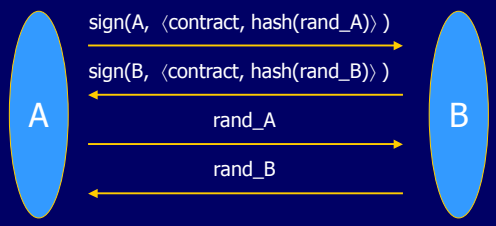


- Trusted third party can force contract
  - Third party can declare contract binding if presented with first two messages.

Commitment (idea from crypto)

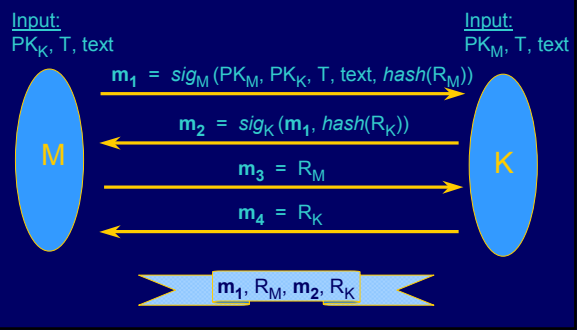
- Cryptographic hash function
  - Easy to compute function f
  - Given f(x), hard to find y with f(y)=f(x)
  - Hard to find pairs x, y with f(y)=f(x)
- Commit
  - Send f(x) for randomly chosen x
- Complete
  - Reveal x

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

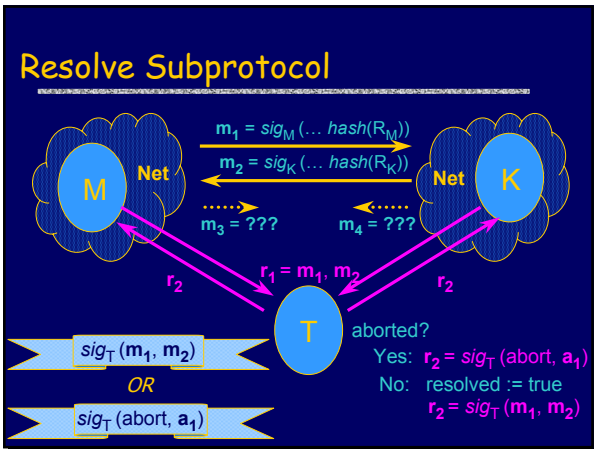
  m_1, R_M, m_2, R_K
- Contract issued by third party

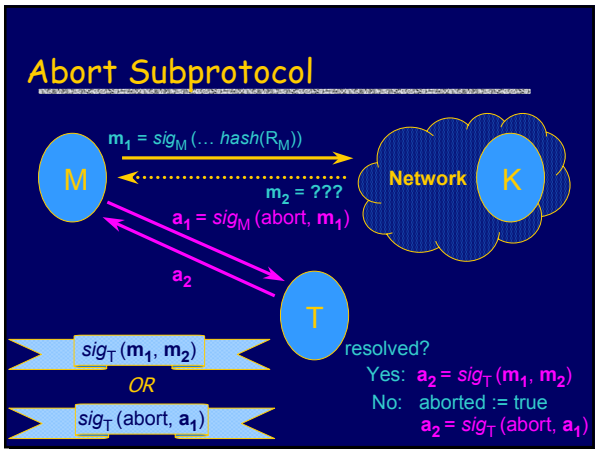
  $sig_T(m_1, m_2)$
- Abort token issued by third party

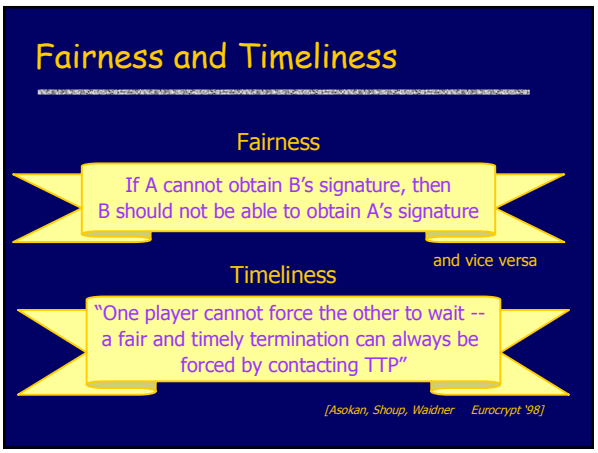
  $sig_T(abort, a_1)$

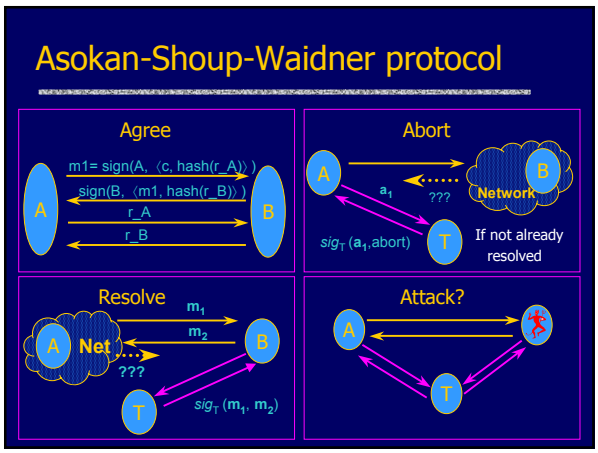
Role of Trusted Third Party

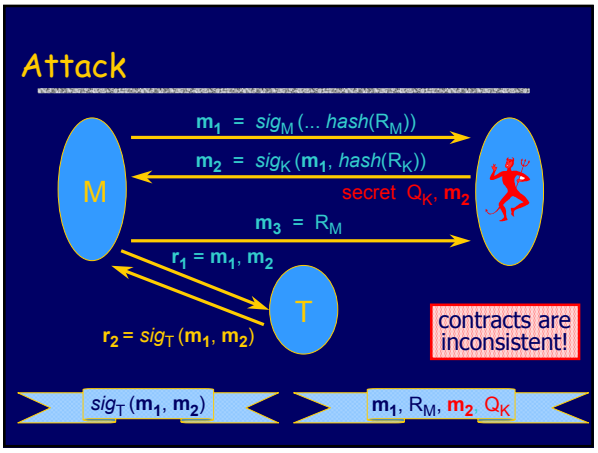
- T can issue a replacement contract
  - Proof that both parties are committed
- T can issue an abort token
  - Proof that T will not issue contract
- T acts only when requested
  - decides whether to abort or resolve on the first-come-first-serve basis
  - only gets involved if requested by M or K

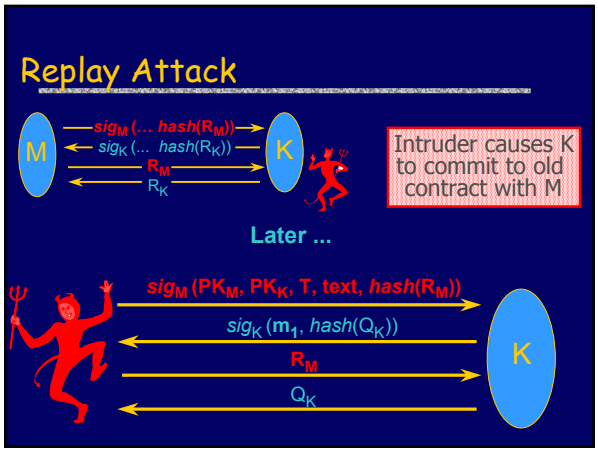

Resolve Subprotocol
M
Net
Net
K
$m_1 = sig_M(\ldots hash(R_M))$
$m_2 = sig_K(\ldots hash(R_K))$
$m_3 = ???$
$m_4 = ???$
r_2
$r_1 = m_1, m_2$
r_2
T
aborted?
Yes: $r_2 = sig_T(abort, a_1)$
No: resolved := true
$r_2 = sig_T(m_1, m_2)$
$sig_T(m_1, m_2)$
OR
$sig_T(abort, a_1)$


Abort Subprotocol
M
Network
K
$m_1 = sig_M(\ldots hash(R_M))$
$m_2 = ???$
$a_1 = sig_M(abort, m_1)$
a_2
T
resolved?
Yes: $a_2 = sig_T(m_1, m_2)$
No: aborted := true
$a_2 = sig_T(abort, a_1)$
$sig_T(m_1, m_2)$
OR
$sig_T(abort, a_1)$

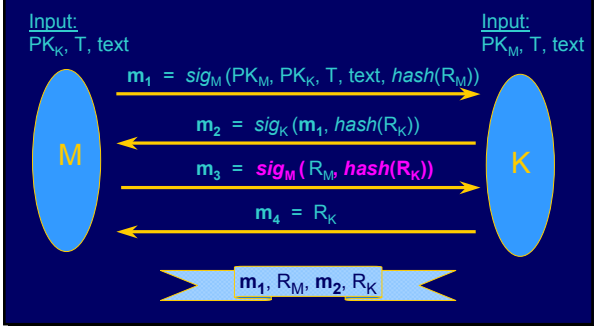

Fairness and Timeliness
Fairness
If A cannot obtain B's signature, then
B should not be able to obtain A's signature
and vice versa
Timeliness
"One player cannot force the other to wait --
a fair and timely termination can always be
forced by contacting TTP"
[Asokan, Shoup, Waidner Eurocrypt '98]


Asokan-Shoup-Waidner protocol
Agree
m1= sign(A, ⟨c, hash(r_A)⟩)
sign(B, ⟨m1, hash(r_B)⟩)
r_A
r_B
A
B
Abort
A
B
Network
a_1
???
$sig_T(a_1, abort)$
T
If not already resolved
Resolve
m_1
m_2
A
Net
B
???
T
$sig_T(m_1, m_2)$
Attack?
A
T


Attack
M
$m_1 = sig_M(\ldots hash(R_M))$
$m_2 = sig_K(m_1, hash(R_K))$
secret Q_K, m_2
$m_3 = R_M$
$r_1 = m_1, m_2$
T
$r_2 = sig_T(m_1, m_2)$
contracts are inconsistent!
$sig_T(m_1, m_2)$
m_1, R_M, m_2, Q_K


Replay Attack
M
$sig_M(\ldots hash(R_M))$
$sig_K(\ldots hash(R_K))$
R_M
R_K
K
Intruder causes K to commit to old contract with M
Later ...
$sig_M(PK_M, PK_K, T, text, hash(R_M))$
$sig_K(m_1, hash(Q_K))$
R_M
Q_K
K

Fixing the Protocol

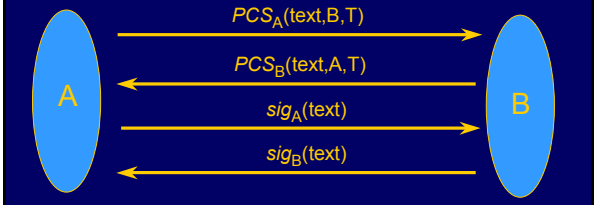


Desirable properties

- 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

Abuse-Free Contract Signing

[Garay, Jakobsson, MacKenzie]



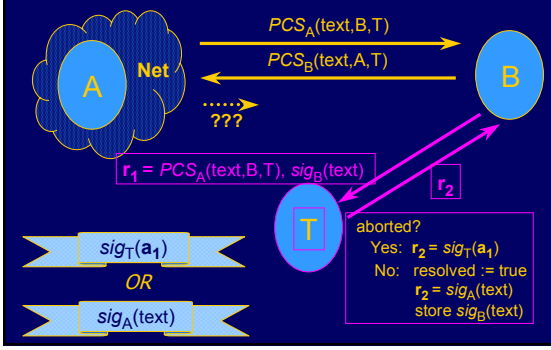
Preventing "abuse" [Garay, Jakobsson, MacKenzie]

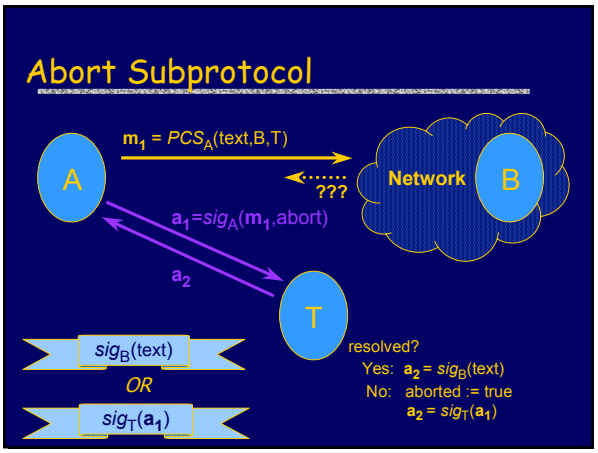
- Private Contract Signature
  - Special cryptographic primitive
  - B cannot take msg from A and show to C
  - T converts signatures, does not use own

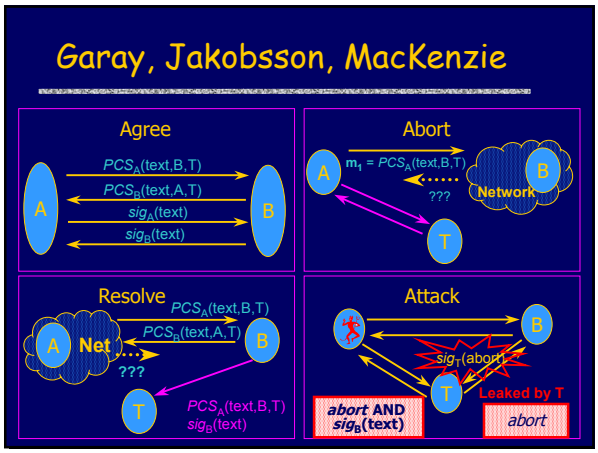
Role of Trusted Third Party

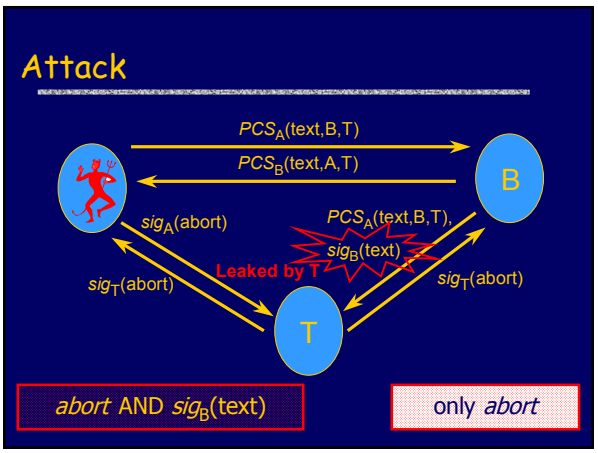
- T can convert PCS to regular signature
  - Resolve the protocol if necessary
- T can issue an abort token
  - Promise not to resolve protocol in future
- T acts only when requested
  - decides whether to abort or resolve on a first-come-first-served basis
  - only gets involved if requested by A or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

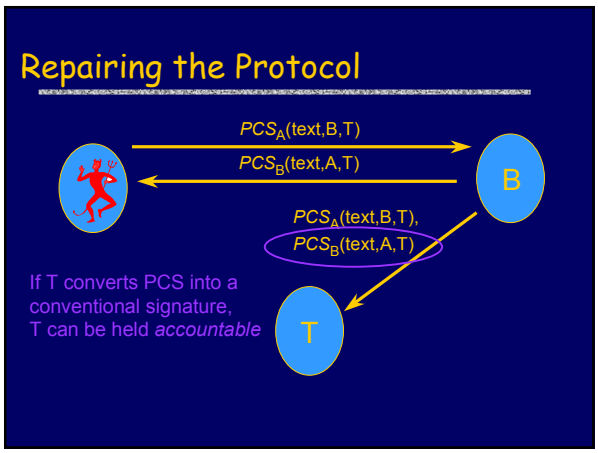
Resolve Subprotocol

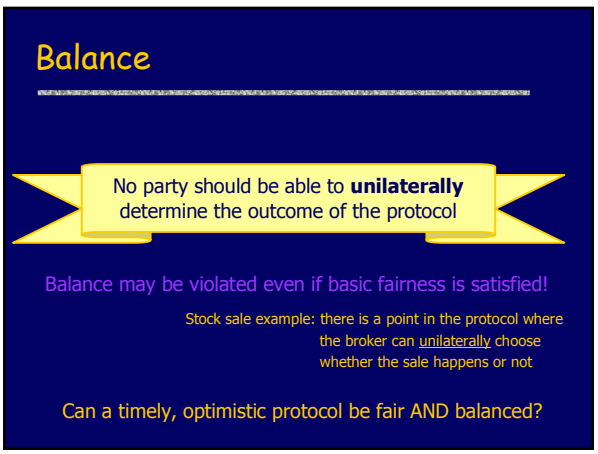


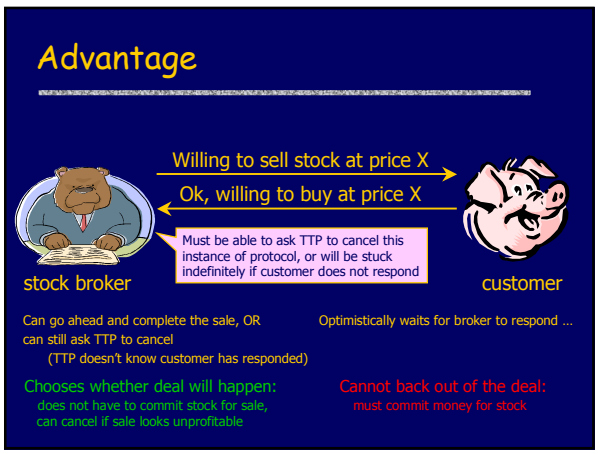

Abort Subprotocol
A
$m_1 = PCS_A(text,B,T)$
???
Network
B
$a_1 = sig_A(m_1,abort)$
a_2
T
$sig_B(text)$
OR
$sig_T(a_1)$
resolved?
Yes: $a_2 = sig_B(text)$
No: aborted := true
$a_2 = sig_T(a_1)$


Garay, Jakobsson, MacKenzie
Agree
$PCS_A(text,B,T)$
$PCS_B(text,A,T)$
$sig_A(text)$
$sig_B(text)$
Abort
$m_1 = PCS_A(text,B,T)$
???
Network
Resolve
$PCS_A(text,B,T)$
$PCS_B(text,A,T)$
Net
???
$PCS_A(text,B,T)$
$sig_B(text)$
Attack
$sig_T(abort)$
abort AND $sig_B(text)$
Leaked by T
abort


Attack
$PCS_A(text,B,T)$
$PCS_B(text,A,T)$
$sig_A(abort)$
$PCS_A(text,B,T)$,
$sig_B(text)$
Leaked by T
$sig_T(abort)$
$sig_T(abort)$
abort AND $sig_B(text)$
only abort


Repairing the Protocol
$PCS_A(text,B,T)$
$PCS_B(text,A,T)$
$PCS_A(text,B,T)$,
$PCS_B(text,A,T)$
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
Willing to sell stock at price X
Ok, willing to buy at price X
Must be able to ask TTP to cancel this instance of protocol, or will be stuck indefinitely if customer does not respond
stock broker
customer
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

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

- 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