CS 395T

Contract Signing Protocols

Real-World Fair Exchange

Immunity deal

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

General Setting

- Two parties agree on the items to exchange, each will release his item if the other releases his
- Physical solution is easy
  - Sit at a table and exchange items simultaneously
- General problem:
  how to exchange information fairly on an asynchronous network?
  - Both parties succeed or both fail

Why is Fair Exchange Difficult?

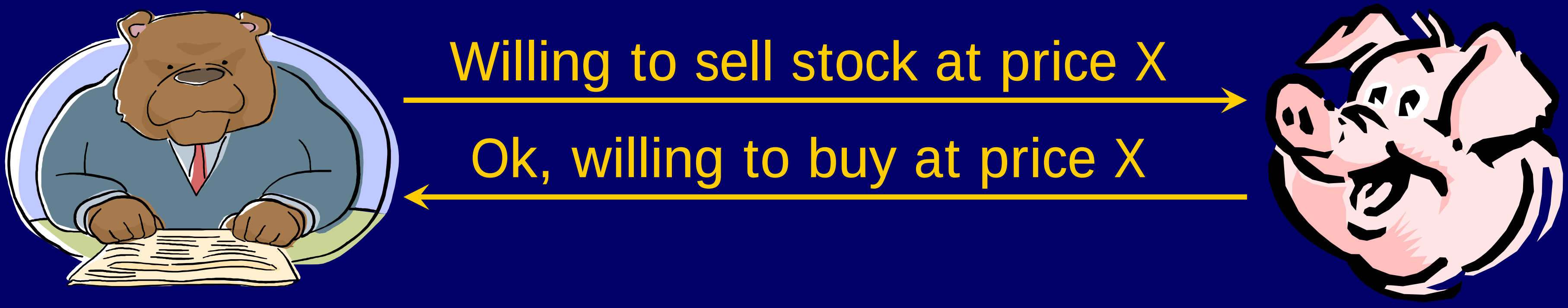
- Cannot trust communication channels
  - Messages may be lost
  - Attacker may insert additional messages
- Cannot trust other party in protocol
  - www.Fly-By-Night.com
  - Public-key certificate does not certify honesty
- There may exist a trustworthy judge or trusted third party
  - Use sparingly, only if something goes wrong, otherwise becomes a communication bottleneck

Focus on Contract Signing Protocols

- Fair exchange of digital signatures
- Two parties want to sign a contract.

  Contract is known in advance to both parties.
  - We'll look at protocols for exchanging signatures, not for contract negotiation (e.g., auctions)
  - Multi-party signing is more complicated
- The attacker could be another party on the network or the person you think you want to sign a contract with
  - In key establishment protocols, usually assume that both parties are honest

Example: Stock Trading



Willing to sell stock at price X

Ok, willing to buy at price X

stock broker

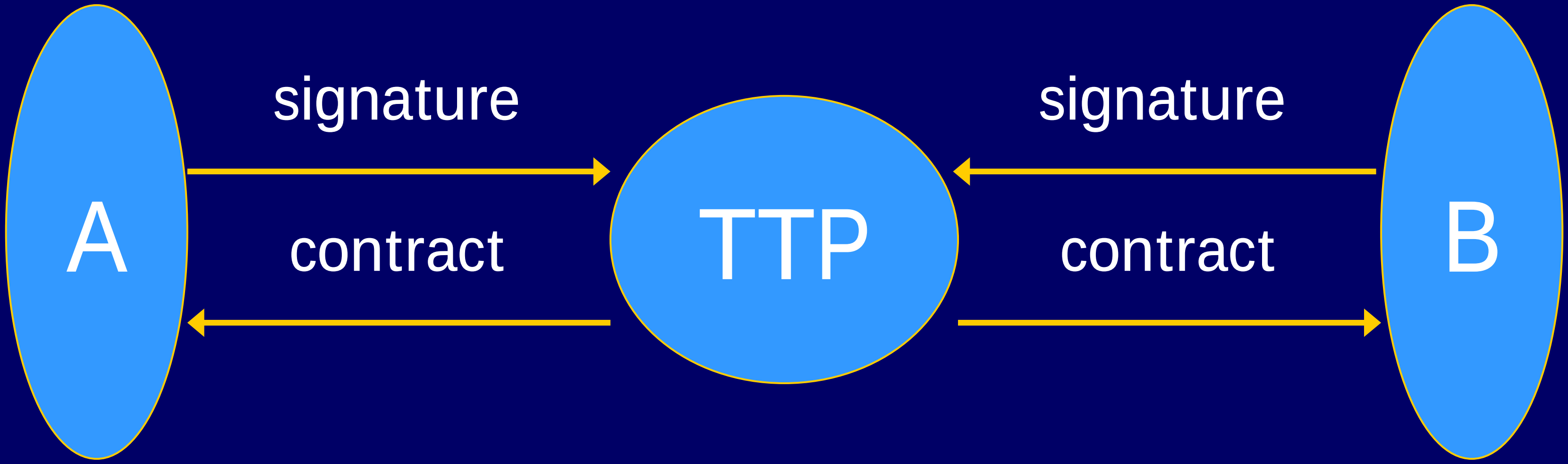
customer

Signed contracts are essential as proofs of agreement in case market price changes

Many Types of Protocols

- Probabilistic protocols
  - We looked at Rabin's and BGMR protocols
- Gradual-release protocols
  - Exchange signatures a few bits at a time
    - Work required to guess remaining bits decreases
    - Main issue: it should be possible to verify that the bits received so far are part of a valid signature
- Fixed-round protocols with trusted third party
  - Impossibility result: no two-party protocol can be fair
    - Reason: fair two-party exchange can be used to solve the distributed consensus problem
  - Need TTP in case one of the parties misbehaves

Contract Signing with Online TTP



Problem: TTP is the communication bottleneck

Can it be removed?

Fundamental Limitation

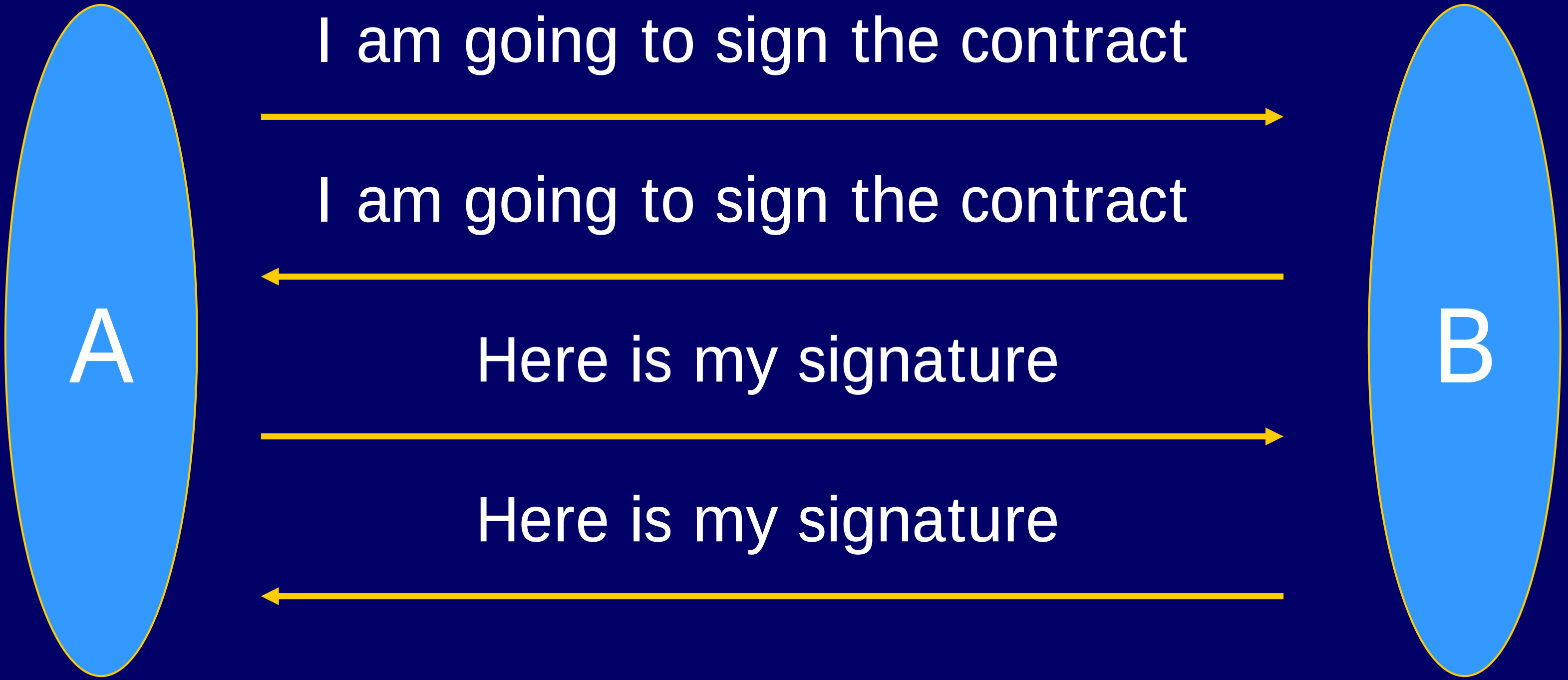
- (Very weak) consensus is not solvable if one or more processes can be faulty
  - *Fisher, Lynch, Paterson. "Impossibility of Distributed Consensus with One Faulty Process". J ACM (1985).*
- Consensus problem in asynchronous setting
  - Several processes want to agree on value of some bit
    - Each process has initial 0 or 1, eventually "decides" on 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

Partial Intuition for FLP Result

- 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.

Optimistic Contract Signing


A
I am going to sign the contract
I am going to sign the contract
Here is my signature
Here is my signature
B

- Involve trusted third party only if something goes wrong
  - Declares contract binding if presented with first two messages

Crypto Magic: Signature Escrows

- Ordinary escrow: $\text{OrdEsc}(\text{sig}_A(m),T)$
  - Similar to $\{\text{sig}_A(m)\}_{pk(T)}$
  - T can extract $\text{sig}_A(m)$ if formed correctly
  - B can’t extract $\text{sig}_A(m)$ and can’t verify what’s inside
- Verifiable escrow: $\text{VerEsc}(\text{sig}_A(m),T)$
  - T can extract $\text{sig}_A(m)$ if formed correctly
  - B can’t extract $\text{sig}_A(m)$ but can verify that A’s signature is inside and that T will be able to extract it

Private Contract Signatures

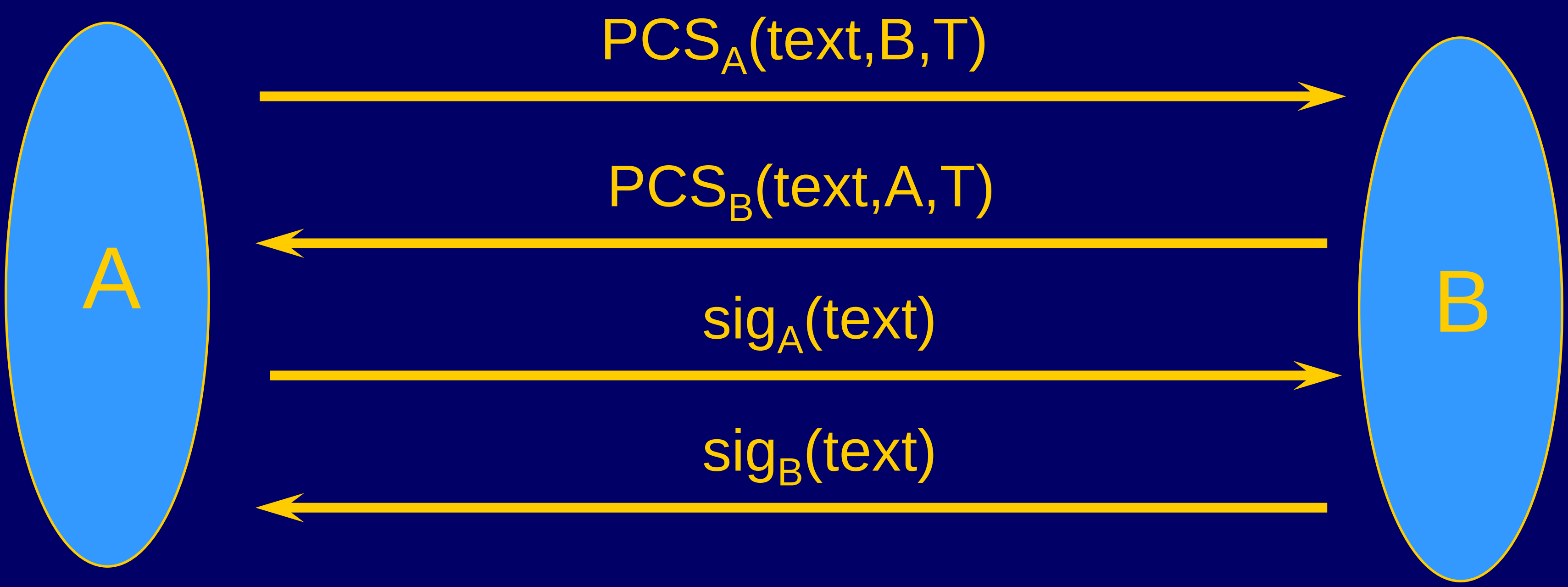
[Garay et al.]

- Private contract signature $PCS_X(m,Y,T)$ is an implementation of verifiable signature escrow
  - Non-interactive zero-knowledge designated-verifier proof of convertible commitment to a signature with a designated converter
- Can be created only by X, but Y can simulate it
  - Therefore, Y cannot use it as proof of X's participation
- T can convert PCS into a universally verifiable signature $sig_X(m)$
  - Y can verify that PCS sent by X can indeed be converted by T into X's signature

Outsider can't distinguish X's private contract signature from Y's simulation

Abuse-Free Contract Signing

[Garay, Jakobsson, MacKenzie]



A

B

$\mathrm{PCS}_A(\mathrm{text},B,T)$

$\mathrm{PCS}_B(\mathrm{text},A,T)$

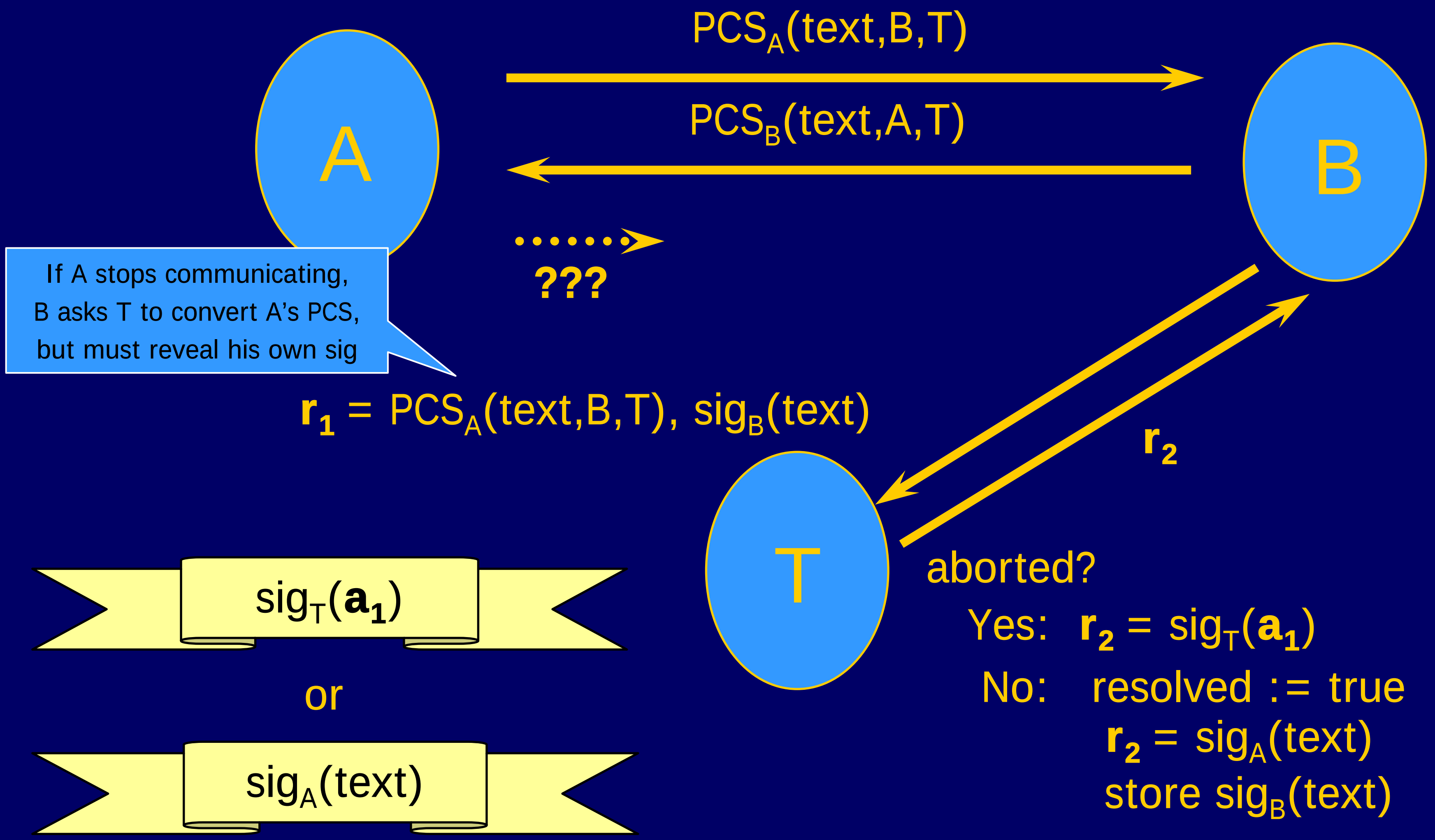
$\mathrm{sig}_A(\mathrm{text})$

$\mathrm{sig}_B(\mathrm{text})$

Role of Trusted Third Party

- T can convert PCS to regular signature (“resolve”)
  - If one of the parties stops communicating, the other party can ask T to convert PCS into signature
- T can issue an abort token (“abort”)
  - Promise not to resolve protocol in future
- T acts only when requested by A or B
  - Decides whether to abort or resolve on a first-come-first-served basis

Resolve Subprotocol



PCS_A(text,B,T)

PCS_B(text,A,T)

???

If A stops communicating,
B asks T to convert A’s PCS,
but must reveal his own sig

$\mathbf{r_1}$ = PCS_A(text,B,T), sig_B(text)

$\mathbf{r_2}$

aborted?

Yes: $\mathbf{r_2}$ = $sig_T(\mathbf{a_1})$

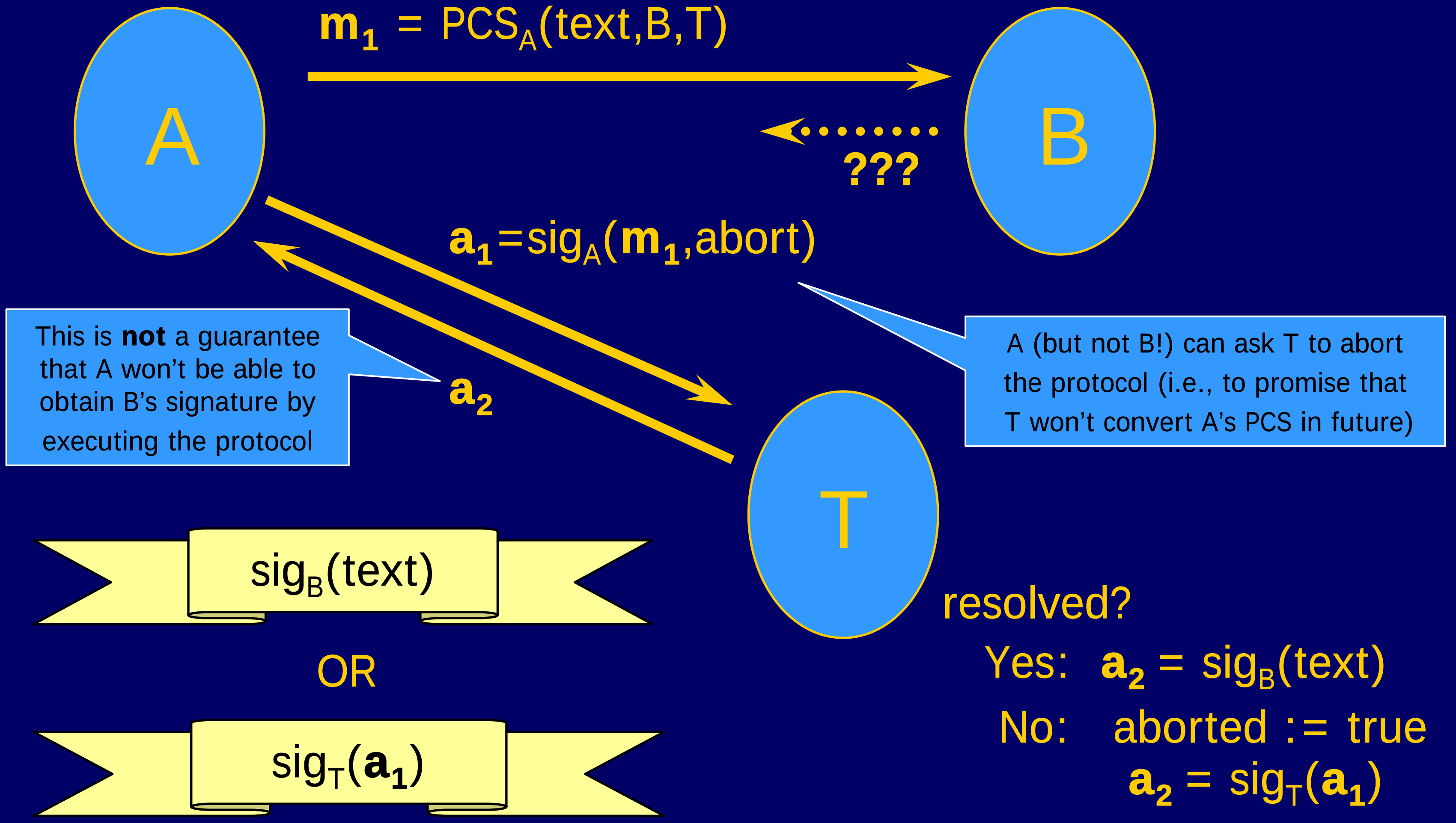
No: resolved := true
$\mathbf{r_2}$ = sig_A(text)
store sig_B(text)

$sig_T(\mathbf{a_1})$

or

sig_A(text)

Abort Subprotocol



$\mathbf{m_1}$ = $\mathrm{PCS_A}$(text,B,T)

$\mathbf{a_1}$=$\mathrm{sig_A}$($\mathbf{m_1}$,abort)

$\mathbf{a_2}$

This is **not** a guarantee that A won’t be able to obtain B’s signature by executing the protocol

A (but not B!) can ask T to abort the protocol (i.e., to promise that T won’t convert A’s PCS in future)

$\mathrm{sig_B}$(text)

OR

$\mathrm{sig_T}$($\mathbf{a_1}$)

resolved?

Yes: $\mathbf{a_2}$ = $\mathrm{sig_B}$(text)

No: aborted := true
$\mathbf{a_2}$ = $\mathrm{sig_T}$($\mathbf{a_1}$)

Desirable Properties

- Fairness
  - Either both A & B get each other’s signature, or none do
- Timeliness
  - Any party can terminate protocol by contacting TTP
- No advantage
  - No party can unilaterally determine the outcome
- No provable advantage
  - No party can prove that it has advantage
- Accountability
  - If a party or TTP cheats, message trace provides evidence of cheating

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

No Advantage (Balance)

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

This property can fail 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?

Example of Advantage

stock broker

Willing to sell stock at price X

Ok, willing to buy at price X

Must be able to ask TTP to “abort” this instance of protocol, or will be stuck indefinitely if customer does not respond

customer

Can go ahead and complete the sale, OR can still ask TTP to “abort”

(TTP doesn’t know customer has responded)

Optimistically waits for broker to respond…

FLP “window of vulnerability” again!

Chooses whether deal will happen:

does not have to commit stock for sale, can cancel if sale looks unprofitable

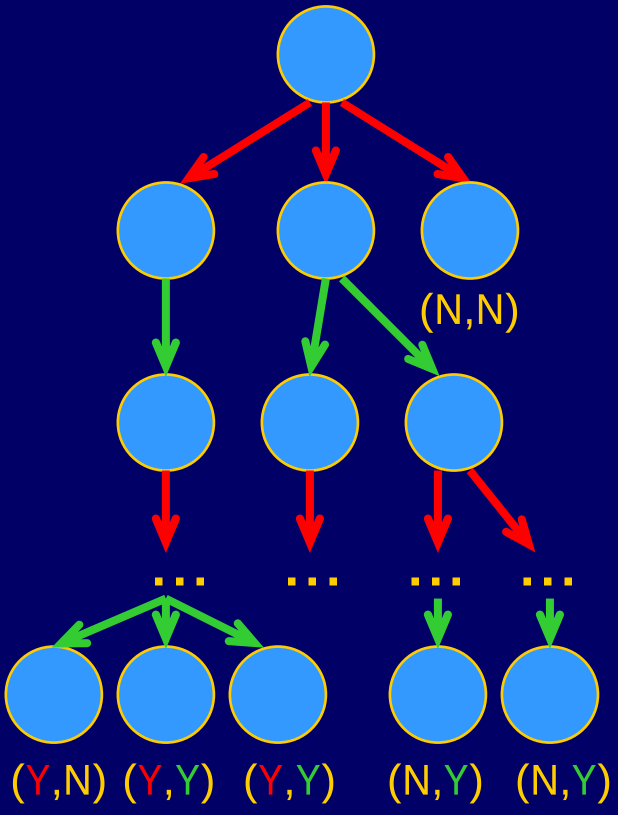
Cannot back out of the deal:

must commit money for stock

Game-Theoretic Model

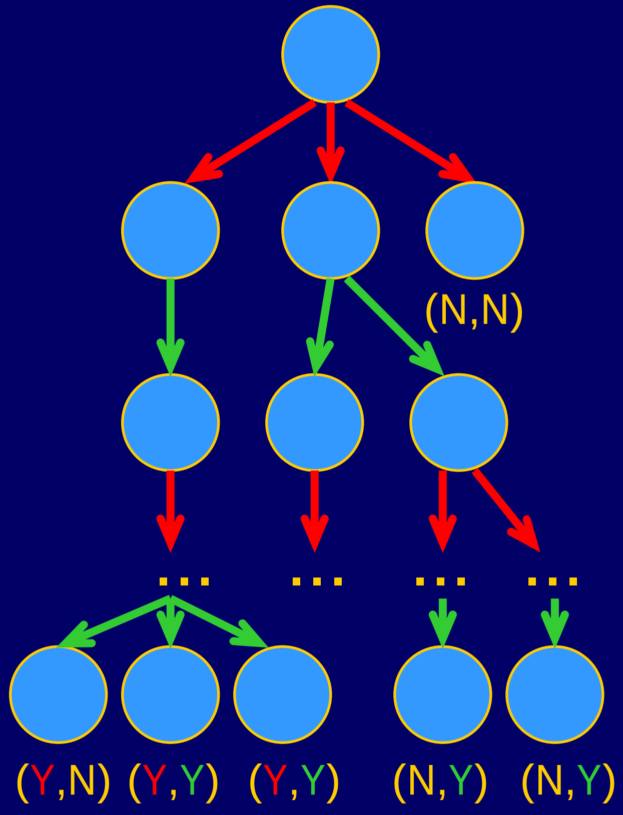
- Each protocol message is a game move
  - Different sets of moves for different participants
- Four possible outcomes (for signature exchange)
  - A has B’s signature, B has A’s signature
  - A has B’s signature, B doesn’t have A’s signature, etc.
- Honest players follow the protocol
- Dishonest players can make any move permitted by the formal model
  - Send any message they can compute
  - Wait instead of responding
- Reason about players’ game strategies

Protocol as a Game Tree



- Every possible execution of the protocol is a path in the tree
- Players alternate their moves
  - First A sends a message, then B, then A …
  - Adversary "folded" into dishonest player
- Every leaf labeled by an outcome
  - (Y,Y) if A has B's signature and B has A's
  - (Y,N) if only A has B's signature, etc.
- Natural concept of strategy
  - Informally, strategy is a rule for responding to any move of the opponent
  - A has a strategy for getting B's signature if, for any move B can make, A has a response move such that the game always terminates in some leaf state labeled (Y,…)

Define Properties on Game Trees



Fairness

No leaf node is labeled (Y,N) or (N,Y)

No advantage (for B)

B never has a strategy to reach (Y,Y)
AND a strategy to reach (N,N)

No provable advantage (for B)

B cannot PROVE that
it has advantage

- Not trace-based properties (unlike secrecy and authentication)
- Very difficult to verify with symbolic analysis or process algebras

Key Idea (omitting many subtleties)

- Define “power” of a signer (A or B) in state s

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

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

Advantage is Unavoidable (Intuition)

- If $\text{Power}_B(s) = 0 \rightarrow \text{Power}_B(s') = 1$ then...
- The move must have been performed by A
  - A must have given B additional information that increased B’s power
- The move by A is not a message to TTP
  - This is an optimistic protocol
- B could abort in state s
  - Follows from timeliness, since B can’t get contract in s
- B can still abort in s’, so B has advantage!
  - Intuition: T doesn’t know that B has received additional information from A, so B can lie to T

Impossibility Result

- Dishonest party has advantage in any fixed-round, timely, optimistic fair exchange protocol
  - Dishonest party always has a strategy for reaching a state where it can unilaterally choose the outcome
  - Similar to FLP impossibility result for consensus
  - Cryptography cannot help
- Bad news for e-commerce
  - Honest party must commit merchandise or money, while dishonest party can still decide whether to go ahead with the deal
  - Need a trusted party in every transaction

“Abuse-Free”: As Good as It Gets

~~No advantage~~ impossible ☹

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

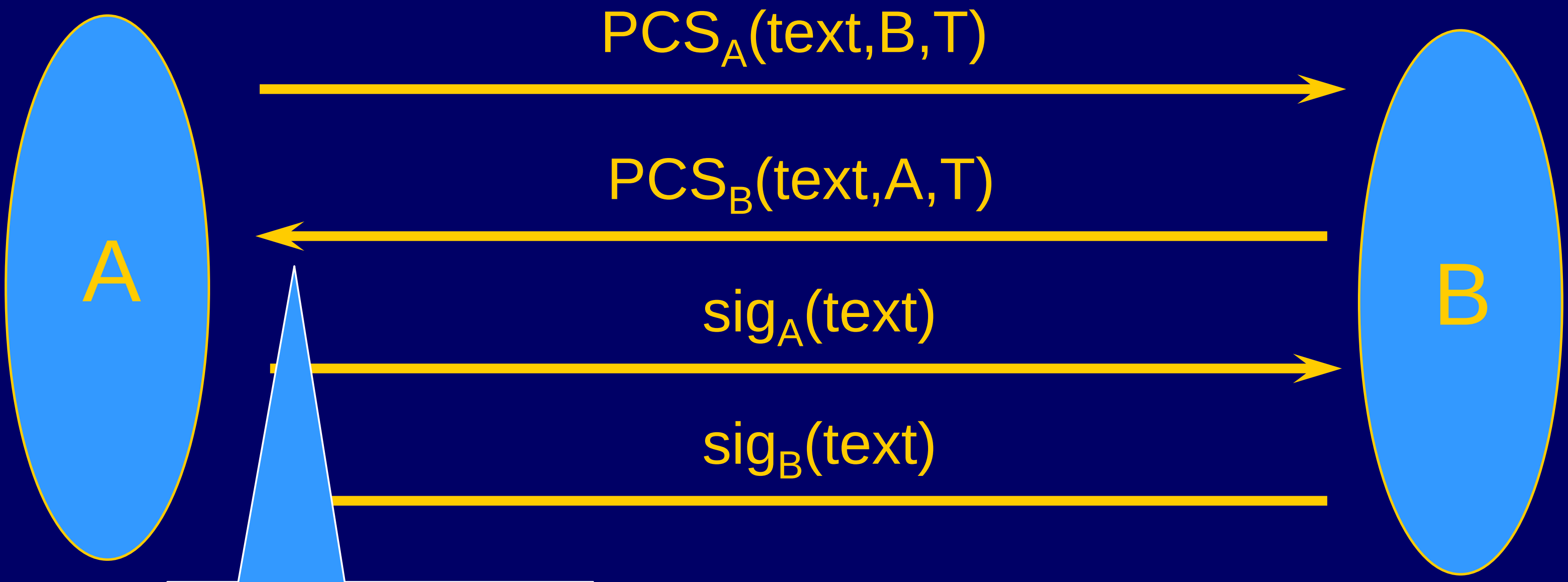
Abuse-Free (No Provable Advantage)

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

Achieved by Garay-Jakobsson-MacKenzie protocol

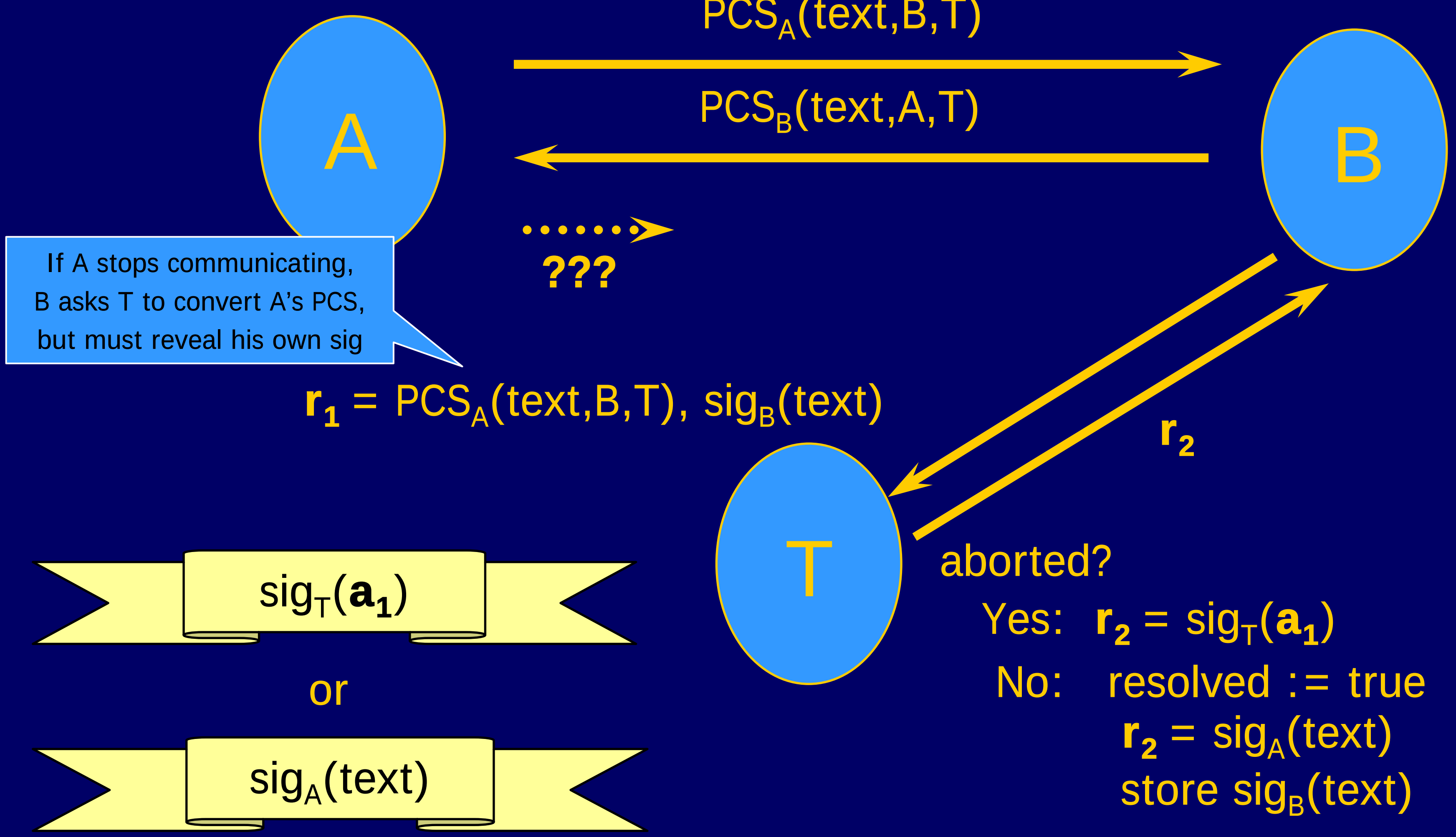
Abuse-Free Contract Signing

[Garay, Jakobsson, MacKenzie]



A has advantage here, but he can't use B's PCS to prove that B is participating (e.g., to solicit another bid)

Resolve Subprotocol



A

B

T

$PCS_A(text,B,T)$

$PCS_B(text,A,T)$

???

If A stops communicating,
B asks T to convert A’s PCS,
but must reveal his own sig

$\mathbf{r_1}$ = $PCS_A(text,B,T)$, $sig_B(text)$

$\mathbf{r_2}$

aborted?

Yes: $\mathbf{r_2}$ = $sig_T(\mathbf{a_1})$

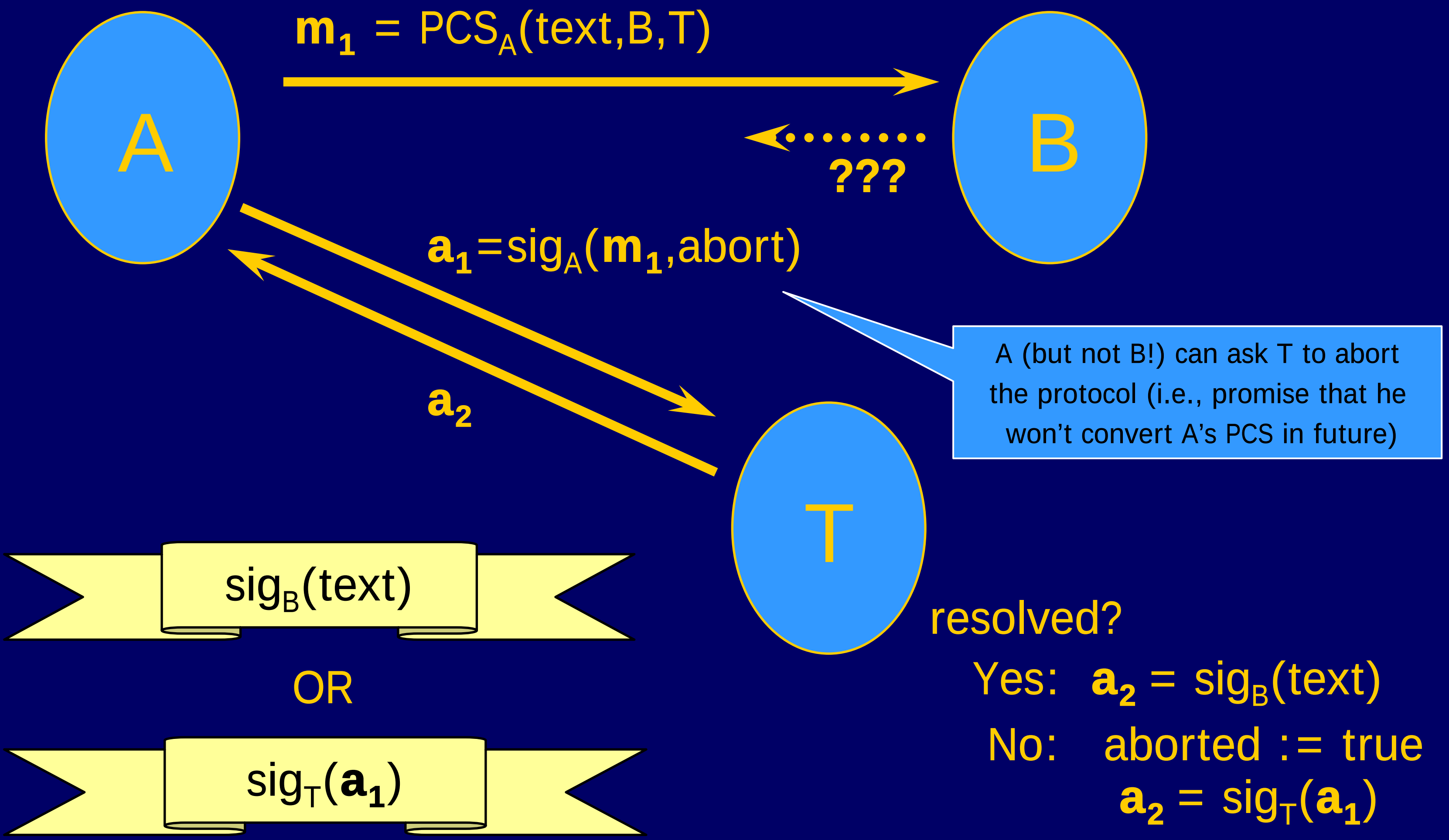
No: resolved := true
$\mathbf{r_2}$ = $sig_A(text)$
store $sig_B(text)$

$sig_T(\mathbf{a_1})$

or

$sig_A(text)$

Abort Subprotocol



$\mathbf{m_1}$ = PCS_A(text,B,T)

$\mathbf{a_1}$=sig_A($\mathbf{m_1}$,abort)

A (but not B!) can ask T to abort the protocol (i.e., promise that he won't convert A's PCS in future)

$\mathbf{a_2}$

sig_B(text)

OR

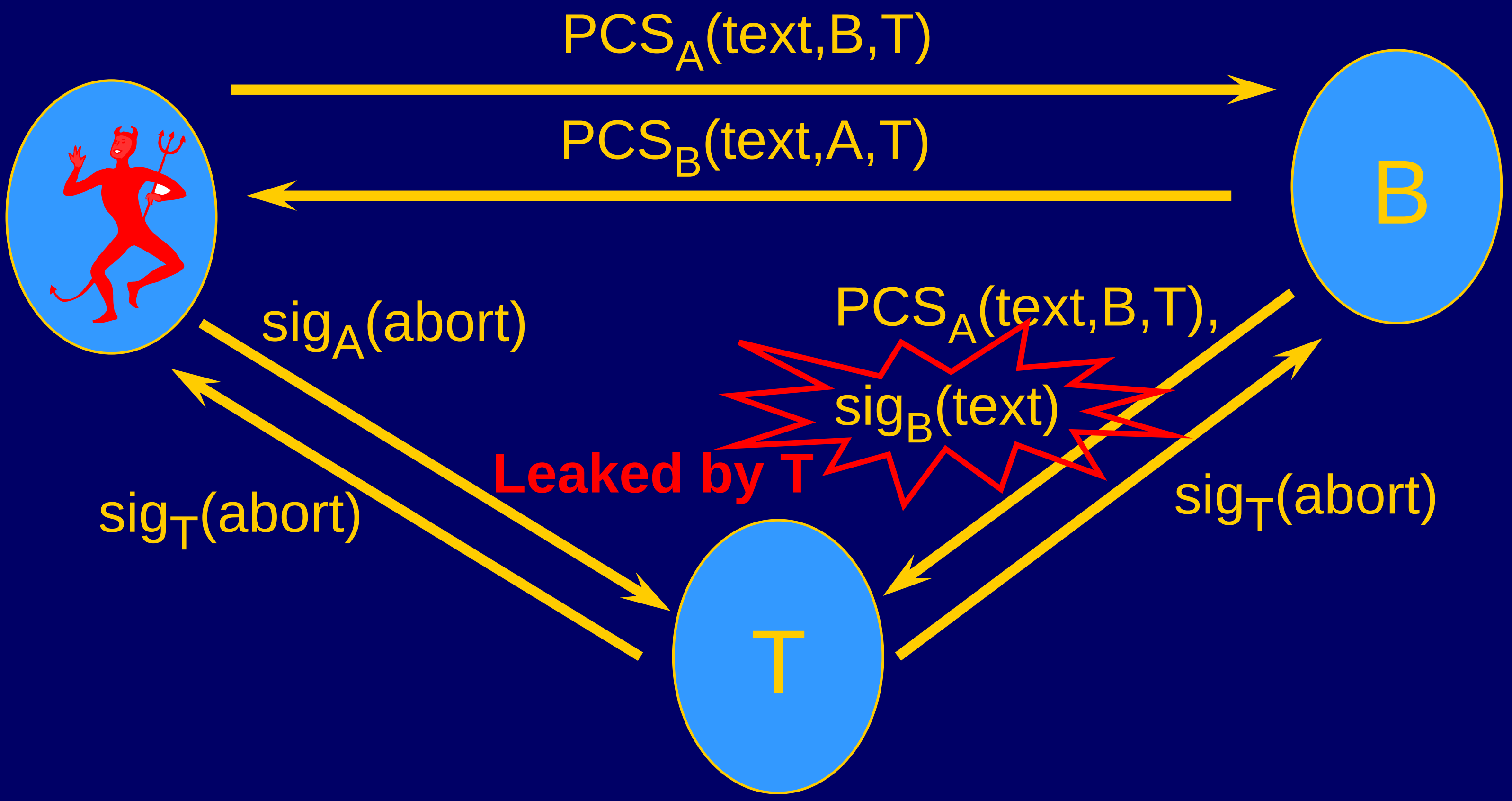
sig_T($\mathbf{a_1}$)

resolved?

Yes: $\mathbf{a_2}$ = sig_B(text)

No: aborted := true
$\mathbf{a_2}$ = sig_T($\mathbf{a_1}$)

Attack on Accountability



PCS_A(text,B,T)

PCS_B(text,A,T)

B

sig_A(abort)

PCS_A(text,B,T),

sig_B(text)

Leaked by T

sig_T(abort)

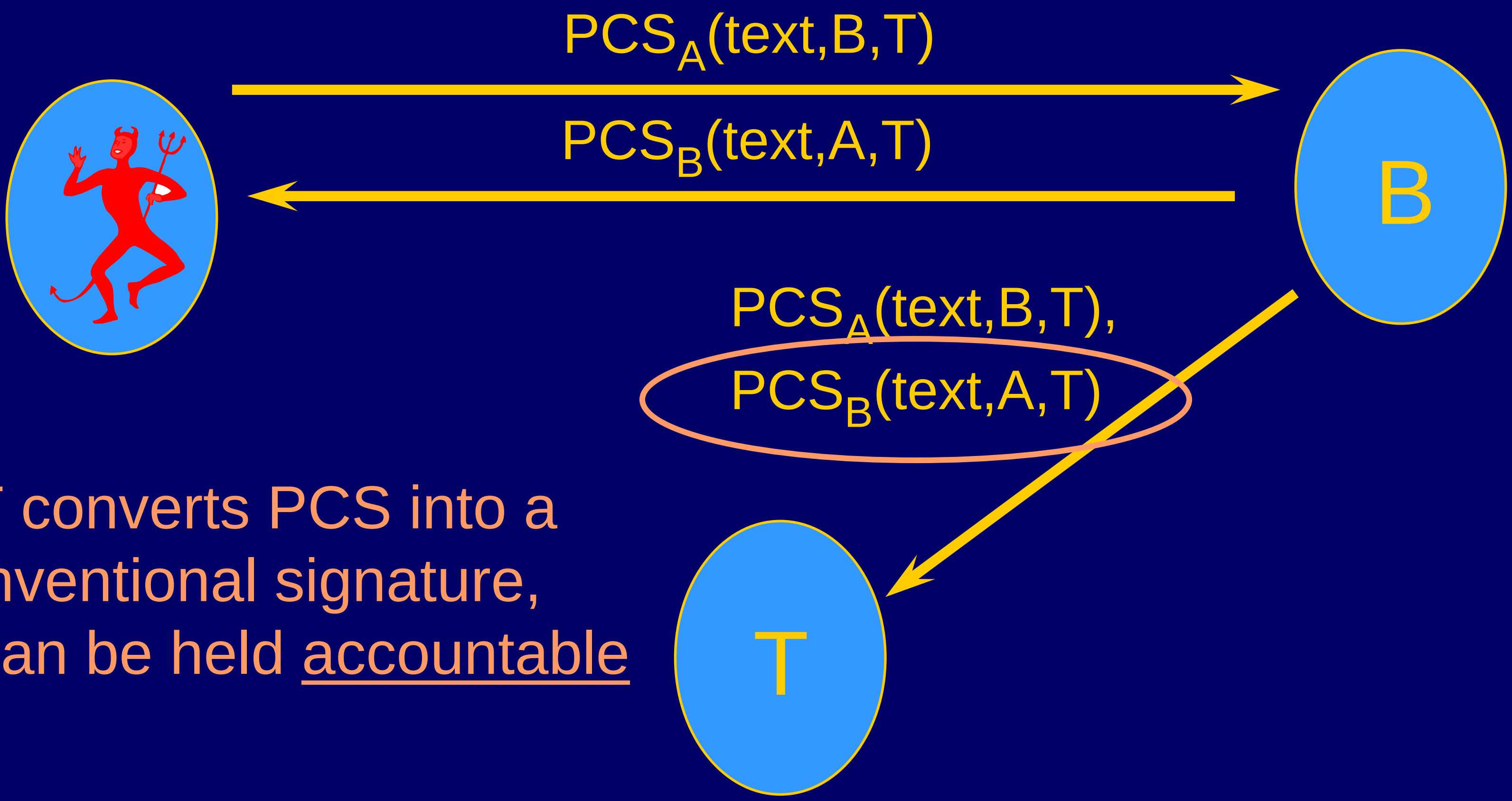
T

sig_T(abort)

sig_T(abort) AND sig_B(text)

only sig_T(abort)

Repairing the Protocol



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