The Problem of Fair Exchange, its Formalization, and its Relation to other Problems in Distributed Computing

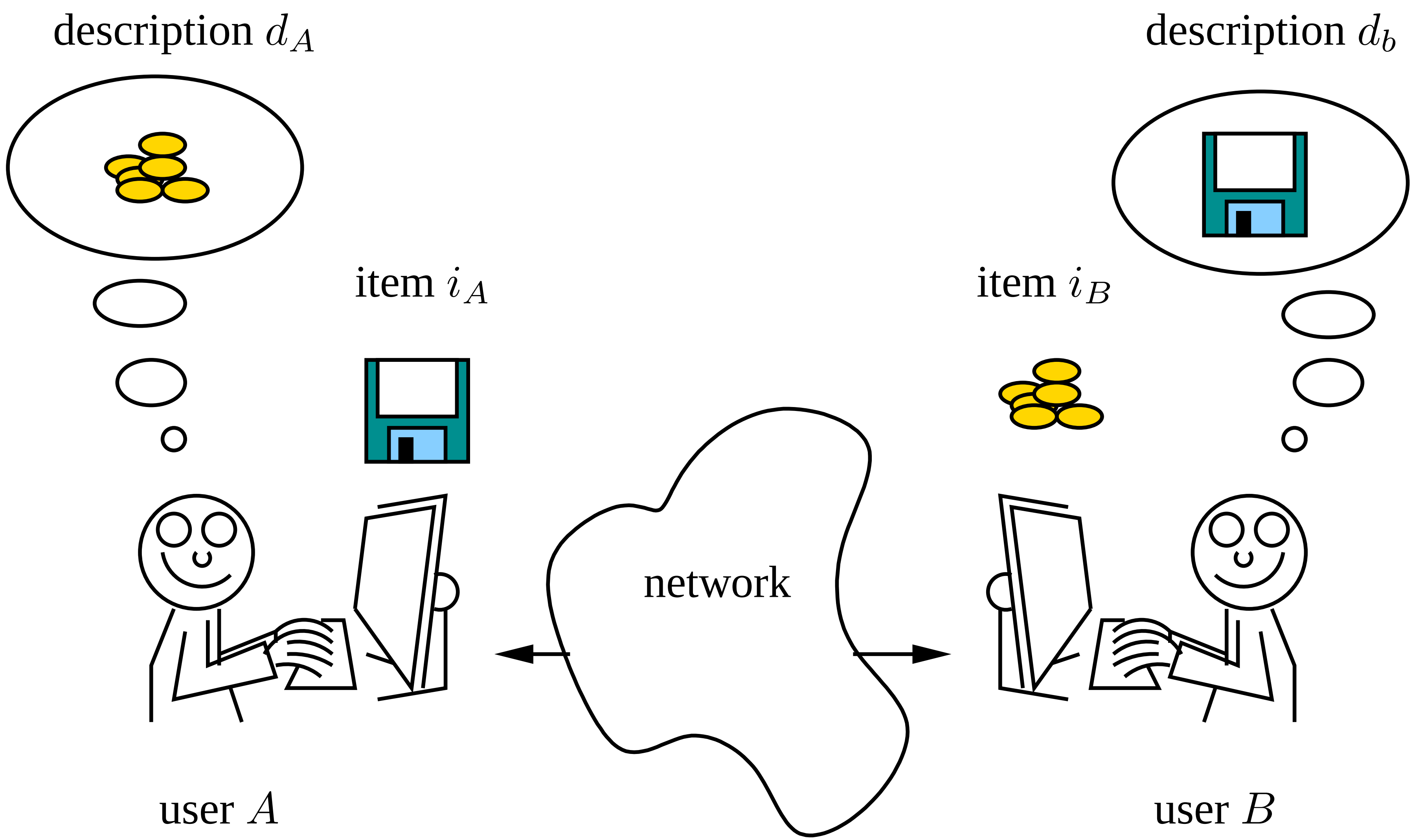
Felix Gärtner

LPD, EPFL, Switzerland

fgaertner@lpdmail.epfl.ch

based on joint work with
Henning Pagnia (BA Mannheim) and Holger Vogt (TU Darmstadt).
See forthcoming article "Fair Exchange" in *The Computer Journal* (Vol. 46, No. 1, 2003).

Exchanging Goods on the Internet



Motivation

- Goal: Exchange the items in a fair manner.
- Fair exchange is an important notion in e-commerce:
  - Exchanging electronic goods and payment.
  - Digital contract signing.
  - Certified e-mail.
  - Mutual disclosure of identities.
- Assumption: items can be fully validated.

Outline

- What is fair exchange (more precisely)?
- Some fair exchange protocols and some impossibilities.
- How formalize fairness?
- Relation to transactions and consensus.
- Some research issues.

Fair Exchange Context

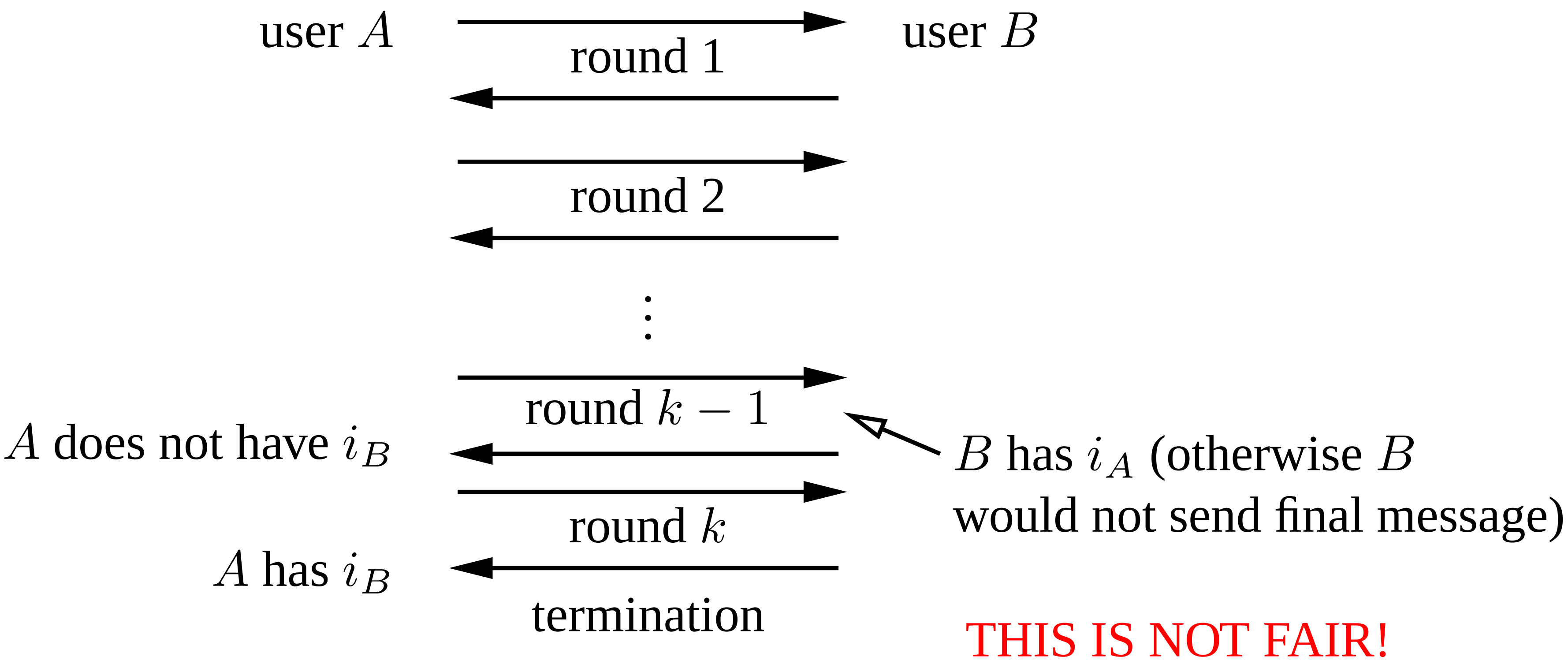
- A, B, i_A, i_B, d_A and d_B are given.
- System model is asynchronous.
  - Users have a weak notion of timeout: A user can unilaterally decide to abandon the exchange (i.e., enforce a termination state for himself).
- Nothing is known about other parties in the system apart from users A and B (for now).
- Adversary assumption:
  - At any point in the protocol, a misbehaving user may go silent and not participate in the exchange anymore.
  - More generally: passive Dolev-Yao model [Dolev and Yao 1983].

Fair Exchange Properties

- A protocol solves fair exchange between two parties A and B if it satisfies three conditions:
  - Effectiveness:
    If both parties behave according to the protocol, both parties do not want to abandon the exchange, and both items match the description then, when the protocol has completed, A has i_B and B has i_B.
  - Termination:
    A party which behaves according to the protocol will eventually complete the protocol (and know that it has completed)
  - Fairness (informal version):
    If at least one party does not behave according to the protocol or if at least one item does not match the description, then no honest participant wins or loses anything valuable.

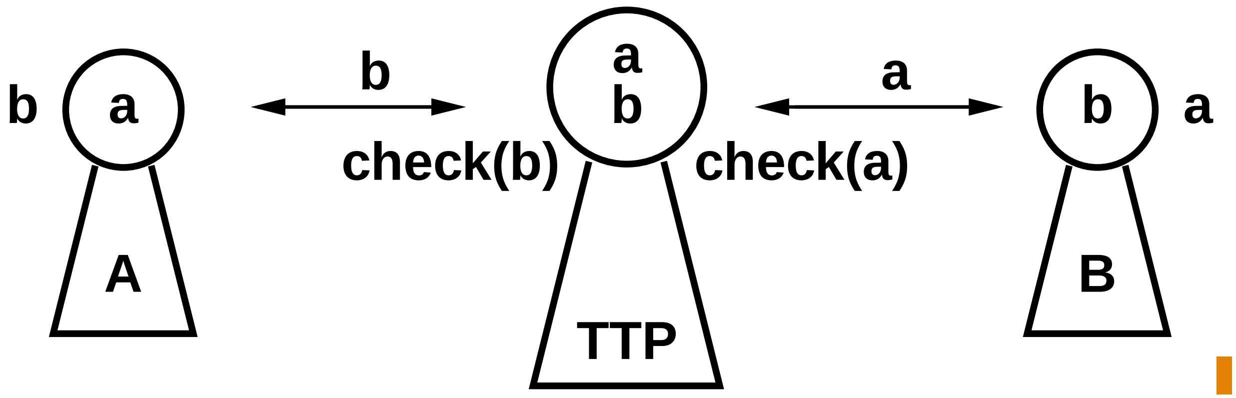
Two-Party Fair Exchange: Impossibility

- In two-party fair exchange, who should go first?
- Impossibility proof by Even and Yacobi [1980]:



Fair Exchange with an Active Trustee

- Simplest protocol: use an active trusted third party (TTP) to perform the exchange.

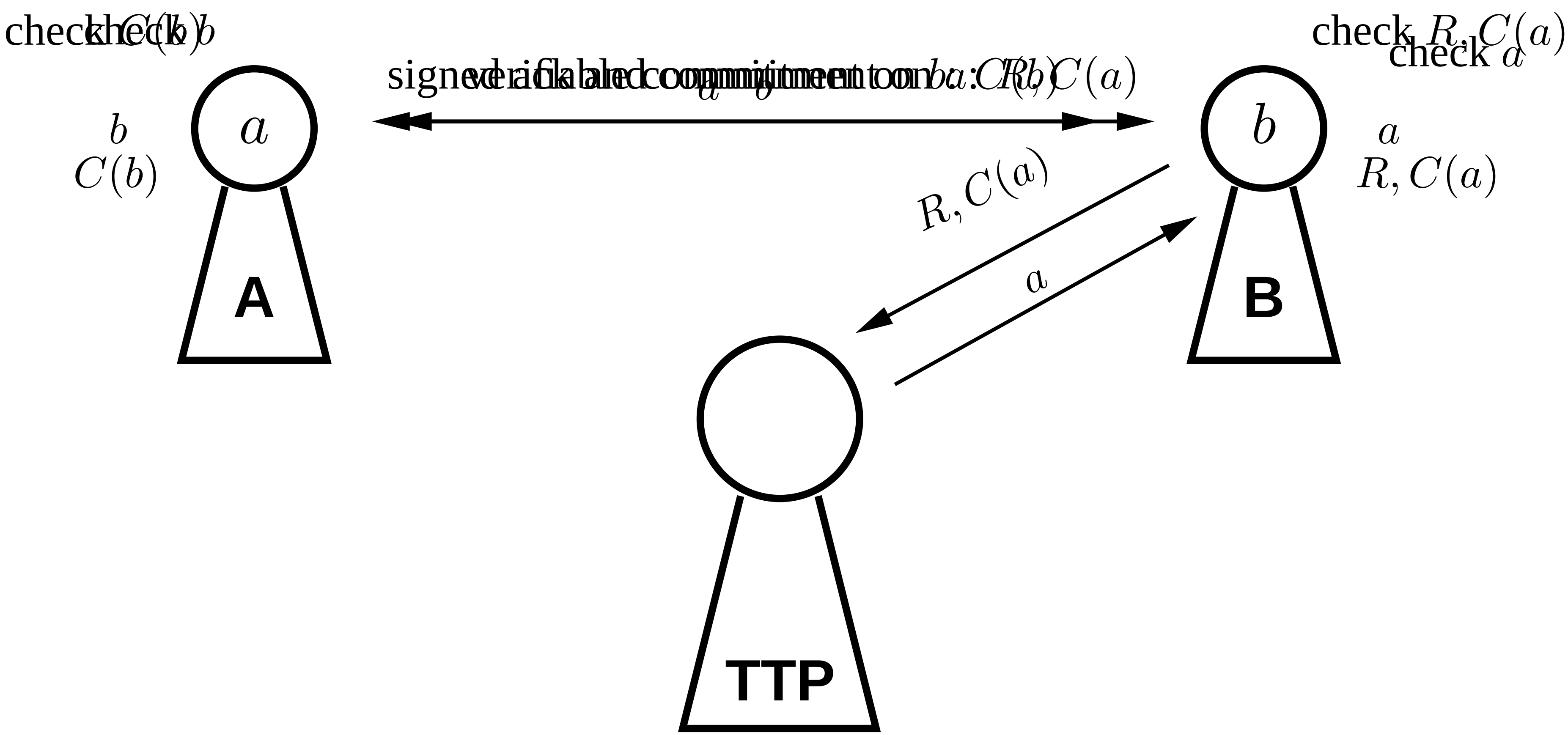


- Validation of fair exchange properties is easy.
- TTP must be trusted and available.

Optimistic Fair Exchange [Asokan et al. 1998]

- TTP can become a bottleneck: only use TTP if something goes wrong.
- Need special item properties to design optimistic protocols:
  - Revocability: One item must be revocable by the TTP (e.g., an electronic payment).
  - Generatability: One item must be generatable by the TTP (e.g., a software package deposited by the TTP).
- Idea: parties send a commitment first, which can be used by the TTP to resolve the exchange in case of a failure.
- Cryptography comes in here:
  - Challenge: how apply cryptography in the right way so that nobody can cheat?

Interaction Pattern of Optimistic Fair Exchange



Why is this fair?

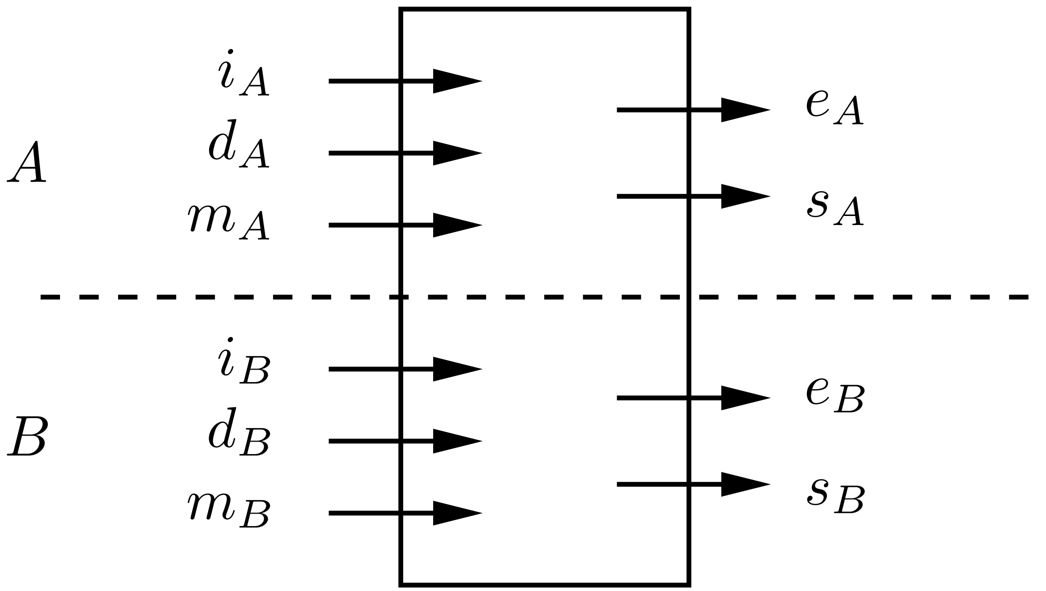
- Two dangerous cases can arise:
  - (shown in previous figure) A may refuse to send a after receiving b:
    * Danger: A has b but B does not have a.
    * B can prove (through R) that B has followed the protocol.
    * TTP can generate a on behalf of B (using $C(a)$).
  - B may refuse to send b after receiving $R, C(a)$:
    * Danger: B has all it needs to resolve the protocol and hence get a.
    * A can request an abort of the exchange at the TTP.
    * Such a request can block the TTP from resolving the exchange for B.
    * In case B was faster, B must deposit b at the TTP, and hence TTP can generate b on behalf of A.

- This is a tricky business and we would like to have some formal methods help increase confidence.

- How formalize fairness?

How Formalize Fairness?

- Let's use trace-based concepts from program verification!



i_x input item
e_x output (exchanged) item
d_x description of desired item
m_x flag indicating malevolence
s_x success/abort indication

- Assume: the protocol ends for a party X by writing something to e_X (initially $\perp$). Can write e_X at most once.

- Malevolent parties "try as hard as they can" before doing this.

Fairness Definitions

- In the following: assume items match description.
- First attempt: fairness as an "always safe" invariant.

$$\Box(e_A = i_B \Leftrightarrow e_B = i_A)$$

  - Problem with atomicity (messages take time).

- Second attempt: fairness as postcondition (based on termination state).

$$\Box[(e_A \neq \perp \wedge e_B \neq \perp) \Rightarrow (e_A = i_B \wedge e_B = i_A)]$$

  - This is the standard approach [Chadha et al. 2001; Shmatikov and Mitchell 2002] in formal verification.
  - Definition depends on the assumptions about misbehaving parties (we require a misbehaving party to do something, it is not an "interface definition").

Fair Exchange vs. Consensus

- Is fair exchange a transaction?
- Define the two-party misbehavior-tolerant consensus problem:
  - Two parties propose a value $v \in \{0, 1\}$ and can decide on a value.
  - If both parties are well-behaved and decide, their decision is the same.
  - The decision value must be a proposed value.
  - Every well-behaving party eventually decides a value.
- Assume we have a primitive

$$e_A := fair\_exchange(i_A, d_A)$$

Can we implement a distributed primitive

$$\delta := consensus(\pi)$$

for two-party consensus?

The Transformation

function $consensus(\pi \in \{0,1\})$ **returns** $\delta \in \{0,1\}$
  local variable $t \in \{0, 1,$ "aborted"$\}$
begin
  $t := fair\_exchange(\pi, desc(\pi))$; {* settings a and b from below *}
  if $t \neq$ "aborted" **then return** π;
  $t := fair\_exchange(\pi, desc(\neg\pi))$; {* settings c and d from below *}
  if $t \neq$ "aborted" **then return** 0; {* or 1 consistently *}
  return π; {* settings e and f from below *}
end

setting	a	b	c	d	e	f
A	1	0	0	1	0	1
B	1	0	1	0	m	m
δ	1	0	0	0	0	1

Fair Exchange vs. Consensus (cont.)

- Hence: Fair exchange is at least as hard to solve as consensus.
  - If consensus is impossible, then so is fair exchange.
- Misbehavior is indistiguishable from a crash (in one instance of fair exchange).
  - Impossibility result of Fischer, Lynch, and Paterson [1985] holds.
  - Corollary: There is no asynchronous two-party fair exchange protocol.
- Same impossibility result as on slide 7, but Even and Yacobi [1980] also cover the synchronous case:
  - There are synchronous two-party consensus protocols, but there are no synchronous two-party fair exchange protocols.
  - Fair exchange seems to be harder than consensus.

Fair exchange vs. Transactions

- Addition of a TTP is a strong assumption which helps make life much easier.
  - Consensus (even Byzantine agreement) is trivially solvable using a TTP.
  - Transformation on slide 15 implicitly postulates a TTP.

- In consensus or atomic commitment protocols there is usually a (distinguished) coordinator process which ensures unanimity.
  - In consensus this can be one of the participating parties.
  - In fair exchange, this must be an external party (secrecy of items must be preserved).

- Intuition: fair exchange is a kind of secure transaction.

Research Issues

- Fair exchange is a good candidate for people coming from the database or consensus world to study security ("the next step").
  - "Secure" consensus is more than Byzantine agreement.
- Asymmetry in fair exchange (due to secrecy).
  - Maybe we need to adapt the consensus definition to be amendable for better comparison?
  - Maybe we need to leave the domain of the usual trace-based formalizations?
- Recent work in the area goes in different directions:
  - Use trusted hardware to implement a low-cost, low-latency TTP [Vogt et al. 2003].
  - Abuse-free fair exchange [Garay and MacKenzie 1999].
  - Formal analysis [Buttyán and Hubaux 2001].

Summary

- Fair exchange is a fundamental buidling block in modern e-commerce.
- Fair exchange is a difficult and costly task since it (usually) involves a (costly) trusted third party.
  - Optimistic protocols help.
- Relation to consensus and transactions does not seem to be entirely clear.

Researchers in consensus: drop everything else and work on this!

Acknowledgements

- Slides produced using pdfLATEX and Klaus Guntermann's PPower4.

References

Asokan, N., Shoup, V., and Waidner, M. 1998. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy* (May 1998), pp. 86–99. Printed version contains some errors. Errata sheet is distributed together with the electronic version.

Buttyán, L. and Hubaux, J.-P. 2001. Rational exchange – A formal model based on game theory. In *Electronic Commerce – WELCOM 2001*, Volume 2232 of *Lecture Notes in Computer Science* (Heidelberg, Nov. 2001), pp. 114–126. Springer-Verlag.

Chadha, R., Kanovich, M., and Scedrov, A. 2001. Inductive methods and contract-signing protocols. In P. Samarati Ed., *Proceedings of the 8th ACM Conference on Computer and Communication Security* (Philadelphia, PA, Nov. 2001), pp. 176–185. ACM Press.

Dolev, D. and Yao, A. C. 1983. On the security of public key protocols. *IEEE Transactions on Information Theory 29*, 2 (March), 198–208.

Even, S. and Yacobi, Y. 1980. Relations amoung public key signature systems. Technical Report 175, Computer Science Department, Technicon, Haifa, Israel.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. 1985. Impossibility of distributed consensus with one faulty process. *Journal of the ACM 32*, 2 (April), 374–382.

Garay, J. A. and MacKenzie, P. 1999. Abuse-free multi-party contract signing. In *Distributed Computing – DISC '99*, Volume 1693 of *Lecture Notes in Computer Science* (Bratislava, Slovak Rep., 27–29 Sept. 1999), pp. 151–165. Springer-Verlag.

Shmatikov, V. and Mitchell, J. C. 2002. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science 283*, 2, 419–450.

Vogt, H., Gärtner, F. C., and Pagnia, H. 2003. Supporting fair exchange in mobile environments. *ACM/Kluwer Journal on Mobile Networks and Applications (MONET) 8*, 2 (April).

Additional Slides

Extendible Consensus

- Adapt definition of consensus to an “extendible” set of processes.

- Define: set of processes $\Pi = \{p_1, p_2, \ldots, p_n\}$.

- Separate Π into Π_a and Π_b such that
  - $\Pi_a \subseteq \Pi$ and $\Pi_b \subseteq \Pi$
  - $\Pi_a \cap \Pi_b = \emptyset$

- Define extendible consensus as follows:
  - For processes in Π_a (uniform) consensus must hold.
  - For processes in Π_b:
    - When a process $p \in \Pi_b$ decides, then this must be the value decided by the processes in Π_a.

- Intuition: processes in Π_b can join in on demand.