Impossibility Results on Fair Exchange

Benoît Garbinato Ian Rickebusch

Université de Lausanne
CH-1015 Lausanne, Switzerland
{benoit.garbinato, ian.rickebusch}@unil.ch

Abstract: The contribution of this paper is threefold. First, we propose a novel specification of the fair exchange problem that clearly separates safety and liveness. This specification assumes a synchronous model where processes communicate by message passing and might behave maliciously. In this model, we prove a first impossibility related to the notion of trust, stating that no solution to fair exchange exists in the absence of an identified process that every process can trust a priori. Finally, we derive an enriched model where processes are divided into trusted and untrusted processes, and we show that an additional assumption is still necessary to solve fair exchange. Intuitively, this result expresses a condition on the connectivity of correct but untrusted processes with respect to trusted processes. We also revisit existing fair exchange solutions described in the literature, in the light of our enriched model, and show that our second impossibility result applies to them.

1 Introduction

In our modern daily lives, the notions of fair exchange and trust are ubiquitous: everyday, without even noticing, we participate in numerous commercial exchanges, which we expect to be fair (and most actually are). A key enabler to make all these exchanges occur is the notion of *trust*. In the physical world, this trust is supported by the identification and the implicit reputation of tangible exchange partners. In the digital world, on the contrary, fair exchange is a surprisingly difficult problem. This can be explained by the lack of trust that characterizes the digital realm. Yet, fair exchange is a fundamental problem that has lately regained interest [AGGV05]. This is partly due to the advent of m-business as a natural evolution of e-business, i.e., extending the possibilities of e-business through the use of mobile devices, e.g., cellular phones.. When it comes to solve fair exchange in such semi-open environments, i.e., where all parties are not necessarily identified a priori, carefully modeling and analyzing trust relationships between processes is a key issue.

Most successful e-business solutions today are following a classical client/server architecture (centralized). This implies that current e-business solutions somehow fail to take full advantage of the Internet's underlying protocols, which were designed to support fully decentralized approaches.[1] For example, current e-business solutions do not provide a

[1] The ARPANET project aimed at building a network with no single point of failure, in order to survive a nuclear strike.

favorable environment for electronic exchanges in the absence of some centralized and trusted server, i.e., they fail to support peer-to-peer only settings. On the other hand, the emergence of mobile devices and ad hoc networks, which often have to operate in a disconnected manner with respect to the Internet, is forcing us to reconsider the current e-business architectures. In that respect, in a companion paper [GR06], we advocate that fair exchange is a key building block when it comes to support peer-to-peer m-business interactions at the middleware level, and we propose such a middleware.

Research on Fair Exchange. Various specifications of the fair exchange problem have been proposed, with slightly different sets of properties [AGGV05, ASW00, Ate99, FR97, Mic03, PG99, RRN05]. Among these properties, fairness is the most difficult to capture and where these specifications tend to differ [RRN05, MGK02]. Most specifications are actually meaningful for exchanges involving only two processes, i.e., they are impossible to satisfy in models allowing more than one Byzantine process. In [AGGV05] for instance, fairness requires that if any correct process does not obtain its item, then no process obtains any items from any other process, which is clearly unsustainable in the presence of two or more Byzantine processes. Indeed, one cannot prevent two Byzantine processes from privately exchanging their items. Our specification of fair exchange, on the contrary, considers the general case where more than two processes might be involved.

Most solutions to fair exchange rely on some kind of Trusted Third Party (TTP). A TTP is a process directly accessible to all processes. Fairness is thus trivially ensured by having processes sending their items to the TTP, which then forwards the items, if the terms of the exchange are fulfilled [BP90]. A TTP brings synchronism and control over terms of the exchange in order to ensure fairness but constitutes a bottleneck and a single point of failure. For this reason, various *optimistic* algorithm have been proposed that only involve the TTP when something goes wrong [ASW00, Mic03, BP90, BWW00]. Other approaches aim at relaxing the assumption made on its trustworthiness. In [FR97] for instance, Franklin and Reiter propose a solution using a *semi-trusted* third party that can misbehave on its own but does not conspire with either of the two participant processes. Departing from the traditional TTP-based approach, some authors proposed solutions based on fully decentralized tamperproof modules [AGGV05, GR06]. Both these solutions assume a network topology where the modules are embedded in their respective processes and cannot communicate directly with one another, however they differ in the power given to the tamperproof modules.

Impossibility Results. Besides proposing a specification or a solution, some authors also discuss the difficulty of fair exchange and propose impossibility results in various models. Oddly enough, it is difficult to find published work on impossibility of fair exchange other than technical reports. Mostly, impossibility results on fair exchange have to be inferred from other impossibilities on problems somehow related to fair exchange, which is far from being straightforward. In [PG99], fair exchange is measured against consensus, and an impossibility result on fair exchange in asynchronous models is shown by comparison with the FLP impossibility [FLP85]. In [EY80], fair exchange is shown to be impossible to solve *deterministically* in an asynchronous system with no Trusted Third Party (TTP). In another feasibility study [KGM95], complex exchanges are broken into

sub-exchanges – each relying on a different TTP – and represented as a graph. Reduction rules are then applied to the graph in order to demonstrate the feasibility of the exchange. This method also makes it possible to illustrate how closely exchange feasibility relies on trust. Another attempt to study what is inherently possible and impossible in safe exchange is proposed in [SW02], using game-theoretic solution concepts. However, the basic exchange model considered in this study only involves two parties, plus a Trusted Third Party. In this paper, we present an impossibility result stating that fair exchange cannot be solved in a synchronous model in the absence of some identified process that every other process can trust *a priori*. We then present another impossibility result in the context of a model with trusted processes.

Contributions and Roadmap. In Sect. 2, we present a general distributed system model with Byzantine failures and we give a fine-grained specification of fair exchange. We also prove that there is no solution to the problem in that model without a notion of trust. In Sect. 3, we modify our model to obtain an enriched model, which can be used as a framework to describe and compare solutions to the fair exchange problem.[2] In this model, we derive necessary conditions for the solvability of the problem. Finally, Sect. 4 revisits existing solutions in the light of our enriched model and points out the common essence of these solutions, while Sect. 5 concludes this paper by giving hints about ongoing and future work.

2 Fair Exchange in the Absence of Trust

We consider a distributed system consisting of a set Π of n processes, $\Pi = \{p_1, \ldots, p_n\}$. Processes are interconnected by some communication network and communicate by message passing. The system is *synchronous*: it exhibits *synchronous computation* and *synchronous communication*, i.e, there exists upper bounds on processing and communication delays. To help our reasoning, we also assume the existence of some global real time clock, whose tick range, noted T, is the set of natural numbers.[3]

Regarding the network topology, we merely assume that processes of Π form a connected graph. Links are reliable bidirectional communication channels, i.e., if both the sender and the receiver are correct, any message inserted in the channel is *eventually* delivered by the receiver, i.e., after some finite amount of time (termination property). A reliable channel also ensures that no message is delivered if it was not previously sent (no creation property).

Executions and Failure Patterns. We define the *execution* of algorithm A as a sequence of steps executed by processes of Π. In each step, a process has the opportunity to atomically perform all three following actions: (1) send a message, (2) receive a message and

[2] As discussed in [GR06], our enriched model can be implemented in practice via dedicated tamperproof chips, already available on the market today.

[3] This global clock is virtual in the sense that processes do not have access to it.

(3) update its local state.[4] Based on this definition, a *Byzantine process* is one that deviates from A in any sort of way, so a Byzantine process is Byzantine against a specific algorithm A. It is a known result that Byzantine failures can only be defined with respect to some algorithm [DGG05]. A *Byzantine failure pattern* f is then defined as a function of T to 2^{Π} where $f(t)$ denotes a set of Byzantine processes that have deviated from A through time t. In a way, a failure pattern f can be seen as a projection of all process failures during some execution of A. Once a process starts misbehaving, it cannot return to being considered correct, i.e., $f(t) \subseteq f(t+1)$. We also define F as the set of all possible failure patterns of A, so $f \in F$.[5]

Let $\mathsf{Byz}(f) = \bigcup_{t \in T} f(t)$ and $\mathsf{Cor}(f) = \Pi - \mathsf{Byz}(f)$ denote respectively the set of Byzantine processes in f and the set of correct processes in f. We define the set F_b of all failure patterns where no more than b processes are Byzantine. More formally, F_b is the largest subset of F such that, for any failure pattern $f \in F_b$, $|\mathsf{Byz}(f)| \leq b$, with $0 \leq b \leq n$:

$$F_b = \{f \in F \ : \ |\mathsf{Byz}(f)| \leq b\} \ with \ 0 \leq b \leq n \ .$$

Note that b is bounded by n, the number of processes in Π. From this definition, we know that b is the maximum number of Byzantine processes in any failure pattern of F_b and that $F_n = F$. Finally, we define the set $F_f^{\sim}$ of all failure patterns producing the same set of Byzantine processes than f. More formally, given some failure pattern f, $F_f^{\sim}$ is the largest subset of F such that, for any failure pattern f' of $F_f^{\sim}$, $\mathsf{Byz}(f') = \mathsf{Byz}(f)$:

$$F_f^{\sim} = \{f' \in F \ : \ \mathsf{Byz}(f') = \mathsf{Byz}(f)\} \ .$$

2.1 The Fair Exchange Problem

The fair exchange problem consists in a group of processes trying to exchange digital items in a fair manner. The difficulty of the problem resides in achieving fairness. Intuitively fairness means that, if one process obtains the desired digital item, then all processes involved in the exchange should also obtain their desired digital item. The assumption is made that each process knows both the set Π of processes participating in the fair exchange and the terms of the exchange. The terms of the exchange are defined by a set D of item descriptions, $D = \{d_1, \ldots, d_n\}$, and a set Ω of pairs of processes (p_i, p_j). Description d_i is the description of the item expected by process p_i. Furthermore d_i is unique, so if $i \neq j$, then $d_i \neq d_j$. A pair (p_i, p_j) defines the receiver p_j of the item offered by p_i. Elements of Ω are defined such that p_j is the image of p_i through a bijective map (or permutation) of Π, with $i \neq j$. Finally, let M denote the set of digital items m_i actually offered by process p_i during an execution of fair exchange, $M = \{m_1, \ldots, m_n\}$. Note that, accordingly, for each description in D there does not necessarily exist a corresponding item in M, since M includes items that might have been offered by Byzantine processes.

[4] At each step, the process can of course choose to skip any of these actions, e.g., if it has nothing to send.

[5] This way of modeling executions and failures is fairly classical [DGG05, CT96].

Fair Exchange as Service. Fair exchange can be seen as a service allowing processes to exchange digital items in a fair manner. Each process offers an item in exchange for a counterpart of which it has the description. The exchange is concluded when every process releases the desired counterpart or the abort item φ, meaning that the exchange has aborted. To achieve this, the service offers the two primitives described below.

> **offer**(m_i, p_j) – Enables the process p_i to initiate its participation in the exchange with processes of Π by offering item m_i to p_j, in exchange for the item matching description d_i, with d_i and Π known a priori.
>
> **release**(x) – Informs the process that the exchange completed and works as a call-back. Process p_i receives item x, which is either the item matching d_i or the abort item φ.

Note that, at the end of an exchange, we say that p_i *releases* an item, meaning that the service calls back the *release* operation of p_i. This convention is similar to the one used for classical *deliver* primitives, e.g., in reliable broadcast [HT93].

Fair Exchange Properties. We now specify the formal properties of the fair exchange problem. While several other specifications exist in the literature [AGGV05, AV03, PG99], our specification differs in that it separates safety and liveness via *fine-grained properties*. Such elemental properties then allow us to better reason on the impossibility to solve fair exchange in various models.

Validity. If a correct process p_i releases an item x, then either $x \in M$ and x matches d_i, or x is the abort item φ.

Uniqueness. No correct process releases more than once.

Non-triviality. If all processes are correct, no process releases the abort item φ.

Termination. Every correct process *eventually* releases an item.

Integrity. No process p_j releases an item m_i, with process p_i correct, if m_i matches description d_k of some correct process p_k, with $p_k \neq p_j$.

Fairness. If any process p_i releases an item m_j matching description d_i, with p_i or p_j correct, then every correct process p_k releases an item matching description d_k.

Among these six properties, the last two, *integrity* and *fairness*, are specific to the problem of fair exchange and define precisely the possible outcomes of fair exchange algorithms. Other specifications of fair exchange usually rely on a single property to capture the notion of fairness [AGGV05, ASW00, PG99]. However we argue that if those specifications are suitable for cases where $n = 2$, they are impossible to satisfy in models allowing more than one Byzantine process. In [AGGV05], for example, the *fairness* property requires that if any correct process does not obtain its item, then no process obtains any items from any other process. This is clearly unsustainable in the presence of two or more Byzantine

processes because one cannot prevent two Byzantine processes from conspiring in order for one of them to obtain the item of the second one. A simple but flawed fix would be to modify the definition as follows: if any correct process does not obtain its item, then no process obtains any items from any *correct* process. If it first seems correct, this definition of *fairness* now allows a correct process to obtain the item of a Byzantine process, even if other correct processes do not obtain anything.

Coming back to our specification, *integrity* ensures that no process obtains an item offered by a correct process and matching the description of a correct process. Notice that this does not prevent a Byzantine process from illicitly obtaining the item destined to or offered by some other Byzantine process, since such a behavior cannot be prevented and does not prejudice any correct process. Then, *fairness* guarantees that if any process obtains its desired item offered by some other process, with at least one of them being correct, then every correct process also obtains its desired item. In other words this property prevents a Byzantine process from taking advantage of a correct process but does not protect other Byzantine processes. More trivially, it also ensures that no correct process takes advantage of any process.

2.2 Impossibility Result

In [EY80], fair exchange is proved to have no solution in an asynchronous model prone to Byzantine failures. In the following, we show that the exchange problem has no deterministic solution even in the context of a perfectly synchronous model, if no complementary trust hypothesis is made. This is the subject of Theorem 1 hereafter.

In order to prove this, we define the notion of *trusted process* as a process that is known to be correct a priori by all other processes. In our model, no such assumption is made about any process of Π, so each process is potentially correct or Byzantine. Now, for sake of simplicity and without loss of generality, we assume that an item is indivisible, i.e., it cannot be sent in pieces. Note that this assumption does not reduce the scope of the impossibility, since allowing an item to be broken into pieces, e.g., using techniques from [Sha79], would result in facing the same fair exchange problem for sending the last piece of item. For the same reason, we assume the item is not encrypted, since having to later exchange the keys in a fair manner in order for the processes to decipher the items would again let us face the same fair exchange problem.

Theorem 1 *In the context of a synchronous model with Byzantine failures, there is no* deterministic *solution to the fair exchange problem, if there is no trusted process, even in the presence of only a single Byzantine process.*

Proof. The proof is by contradiction.
Assume that some algorithm A solves fair exchange and that there are no trusted process. Consider an execution E of A in which all processes are correct. From the *non-triviality*, *termination* and *validity* properties of FE, in E, every process releases its desired item, and in particular, some process p_i releases item m_j, with m_j matching description d_i and

$(p_j, p_i) \in \Omega$, and some process p_k releases item m_i, with m_i matching description d_k and $(p_i, p_k) \in \Omega$. Now since no process can be trusted and Byzantine processes cannot be detected, in any execution, no process other than p_i and p_k may hold item m_i. So we know that in a previous step of E, p_k receives m_i from p_i. We now consider the two following cases: either (a) p_i sends m_i after receiving m_j or (b) p_i sends m_i before receiving m_j.

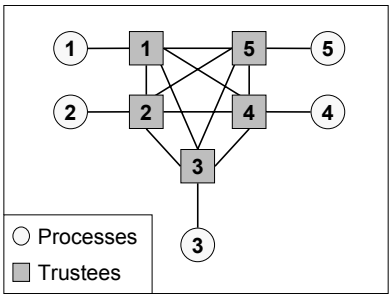
Case (a): Since there is no trusted process, if p_i sends m_i after receiving m_j, we can derive an execution E', similar to E, in which p_i is Byzantine and deviates from A after receiving m_j by omitting to send m_i and by releasing m_j. Since no process is trusted, in E', no process other than p_i holds m_i. So from the *no creation* property of reliable channels, p_k never receives and thus never releases m_i. To satisfy the *validity* and *termination* properties, in E', p_k releases φ but thus violates *fairness*. So, in E, p_i does not send m_i after receiving m_j. Furthermore, this is true for every process, so from the definition of Ω and by circular reasoning, in E, all items are sent at the same time. This now leaves us with case (b).

Case (b): We know that, in E, p_i sends m_i before receiving m_j and that all items are sent at the same time. Now, since there is no trusted process, we can derive an execution E'', similar to E, in which p_j is Byzantine and deviates from A by omitting to send m_j. Since all items are sent at the same time, p_j receives and releases some item m_x matching d_j. Since no process is trusted, in E'', no process other than p_j holds m_j. So from the *no creation* property of reliable channels, p_i never receives and thus never releases m_j. To satisfy the *validity* and *termination* properties, in E'', p_i releases φ but thus violates *fairness*. So finally, Algorithm A does not solve fair exchange. A contradiction. □

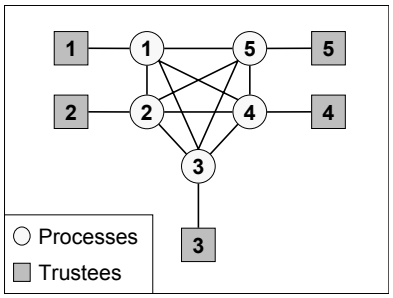
3 Fair Exchange in the Presence of Trust

As described in Sect. 2, we consider a distributed system consisting of a set Π of n *processes*, $\Pi = \{p_1, \ldots, p_n\}$. Processes of Π are also called *participants*. We complete our model with a set Π' of n trusted processes, $\Pi' = \{p'_1, \ldots, p'_n\}$, i.e., a *trusted process* is known a priori to be correct by all other processes. Processes of Π' are called *trustees* or *trusted* processes. Note that, hereafter, to avoid confusion, the term *process* will only be used to describe participants, unless clearly mentioned otherwise. Furthermore, each trustee p'_i is matched in a one-to-one relationship with the corresponding participant p_i and is directly connected to it. Π^+ is then the set of all $2n$ participants and trustees, i.e., $\Pi^+ = \Pi \cup \Pi'$. As illustrated in Fig. 1, processes and trustees are interconnected by a communication network and no additional assumption is made about the network topology, other than the fact that it is a connected graph. Participants are process actually taking part in the exchange by offering and demanding items, and they may exhibit Byzantine behaviors. Trustees on the contrary are *trusted processes* that have no direct interest in the exchange. Intuitively, the role of a trustee is to decide when it is appropriate to provide its matched participant with its expected item.

As we shall see in Sect. 4, this model allows to describe and compare various solutions



(a) TTP-based topology.



(b) Guardian angels topology.

Figure 1: Examples of valid topologies as defined in our model.

proposed in the literature. Simpler topologies, using less trustees and hence some of them matched with more than one participant, can easily be transformed to fit our model. Indeed, any trustee matched with several participants can be represented by a cluster of as many trustees, fully interconnected, and with each trustee matched with one participant. For example, in the case of the Trusted Third Party (TTP), the TTP can be transformed into a cluster of n fully interconnected trustees.

3.1 Impossibility Result in the New Model

In Sect. 2.2, we have shown that without some notion of trust, one cannot solve fair exchange. On the other hand, we know that a cluster of trustees acting as a TTP yields a solution. However, depending on the network topology, the presence of trustees is not sufficient. In the following, we show that in the context of the enriched model, a necessary condition to solve fair exchange is to have every correct participant reliably connected to a majority of trustees. This is the subject of Theorem 2 given hereafter and which relies on Lemma 1.

Beforehand, we need to define the notion of *reliable path* as follows. Let p_i and p_j be two correct processes of Π_+ (either processes or trustees). We say that p_i and p_j are connected through a *reliable path*, if there exists at least one path between p_i and p_j such that, either p_i and p_j are directly connected or p_i is directly connected to a *correct* process $p_k \in \Pi_+$ and there exists a *reliable path* between p_k and p_j. Furthermore, given a process $p_i \in \Pi$ (participants) and a failure pattern $f \in F$, we define $C^f_{p_i}$ as the largest subset of Π' (trustees) such that any trustee p'_j of $C^f_{p_i}$ is connected through a reliable path to p_i. Finally, we define the *reachable majority* condition as the condition under which, for any correct process $p_i \in \Pi$ and any failure pattern $f \in F_b$, $|C^f_{p_i}| > \frac{n}{2}$, even in the presence of up to b Byzantine processes. Intuitively, this means that, even in the worst case scenario, p_i is connected through a reliable path to a majority of trustees, i.e., $\lfloor \frac{n}{2} + 1 \rfloor$. Note that the *reachable majority* condition is a connectivity condition that translates into a maximum

number of Byzantine processes that a certain model of network topology is capable of sustaining. This point is further illustrated in Sect. 4.

Lemma 1 *If an algorithm A solves fair exchange with up to b Byzantine processes, then for any failure pattern $f \in F_b$, there exists an execution associated with a failure pattern $f' \in F_f^{\sim}$ such that every process of $\mathsf{Cor}(f')$ releases its expected item.*

Proof. Consider an execution E of A in which all processes are correct, from the *non-triviality, termination* and *validity* properties, in E every process releases its correct item. Now consider any failure pattern $f \in F_b$. From E, we derive an execution E' associated with a failure pattern $f' \in F_f^{\sim}$, i.e., $\mathsf{Byz}(f') = \mathsf{Byz}(f)$, such that, in E', every process of $\mathsf{Byz}(f')$ deviates from A just before releasing its item, e.g., by crashing. Since E' is indistinguishable from E for all correct processes, every process in $\mathsf{Cor}(f')$ releases its correct item.

Theorem 2 *In the context of a synchronous model with trustees and Byzantine failures, there is no deterministic solution to the fair exchange problem, if the reachable majority condition is not satisfied, even in the presence of a single Byzantine process, i.e., $b = 1$.*

Proof. The proof is by contradiction.
Assume that some algorithm A solves fair exchange and that the *reachable majority* condition is not satisfied, i.e., there is some correct process $p_i \in \Pi$ and some failure pattern $f \in F_b$ for which $|C_{p_i}^f| \leq \frac{n}{2}$, even for $b = 1$. From Lemma 1, we know that there exists an execution E' associated to a failure pattern $f' \in F_f^{\sim}$, such that every process in $\mathsf{Cor}(f')$ releases its expected item. Hence, in E', p_i receives its expected item, e.g., m_j, from its trustee p_i'. We now have to consider the two following cases: (a) the transmission of m_j from p_i' to p_i depends on the reception by p_i' of some message x sent by some trustee $p_j' \in \Pi' - C_{p_i}^f$, and (b) the transmission of m_j from p_i' to p_i is independent of the reception by p_i' of any message sent by any trustee $p_j' \in \Pi' - C_{p_i}^f$.[6]

Case (a): We can derive an execution E'', similar to E', where message x is blocked by some Byzantine process along the *unreliable* path between p_i' and p_j', as well as any following messages. Since E'' is indistinguishable from E' for any process unreliably connected to p_i', i.e., processes associated with trustees of $\Pi' - C_{p_i}^f$, these processes thus release their expected item in E''. However, in E'', p_i' never receives x. Since the transmission of m_j depends on the reception of x, p_i' never sends m_j to p_i. To satisfy the *validity* and *termination* properties, p_i releases φ but thus violates *fairness*. This leaves us with case (b).

Case (b): We can derive an execution E''', similar to E', in which some Byzantine process p_k fails to send the expected item to some trustee $p_j' \in \Pi' - C_{p_i}^f$, with p_j correct and $(p_k, p_j) \in \Omega$. Since the transmission of m_j from p_i' to p_i is independent from the reception of any message sent by any trustee of $\Pi' - C_{p_i}^f$ (included p_j'), for p_i' and

[6] Note that by definition, $C_{p_i}^f = C_{p_i}^{f'}$, for any failure f and f' such that $f' \in F_f^{\sim}$.

p_i, executions E''' and E' are indistinguishable. So, in E''', p_i releases its expected item. However, since p'_j never receives the expected item of p_j, neither does p_j. To satisfy the *validity* and *termination* properties, p_j eventually releases φ but thus violates *fairness*. So algorithm A does not solve fair exchange. A contradiction. □

4 Revisiting Existing Solutions

Trusted Third Party (TTP). Several algorithms described in the literature rely on the TTP paradigm. The simplest TTP-based algorithm consists in having processes send there items to a centralized trustee, the TTP. The TTP verifies that the terms of the exchange are respected and, if this is the case, forwards the items. These TTP-based solutions naturally fit in our enriched model. Our model uses n trustees compared to a unique one in TTP-based solutions. Mapping the TTP model to our model is done by having all n trustees jointly play the role of the TTP by forming a cluster of fully interconnected trustees. The network topology of this solution is such that each process is directly connected to one distinct trustee of the cluster. It is then fairly obvious that the TTP topology is so secure that the reachable majority condition is satisfied for any number of Byzantine processes. Figure 1(a) shows an example of the TTP topology with five processes.

Guardian Angels. In [AGGV05, AV03], guardian angels are defined as tamperproof security devices that are considered correct. Processes are fully interconnected by a communication network with bidirectional reliable channels. There are n guardian angels but each of them is only connected to one process. In other words, each process can directly communicate with its assigned security device but needs to go through some untrusted process to communicate with other security device. Intuitively, in order to solve fair exchange, each item is encrypted and sent to the security device of the corresponding process, i.e., the process expecting the item. Security devices then enter a synchronization protocol, which upon success enables the devices to send the items to the processes. The assumption is made that the security devices are able to check the validity of the items and to encrypt messages. In a model with no upper bound on the number of Byzantine processes, the solution given solves fair exchange with a certain probability. The authors of [AGGV05, AV03] also show that, even in a synchronous model with security devices, no deterministic algorithm solves fair exchange without an honest majority, i.e., without $b < \frac{n}{2}$.

The guardian angels approach fits our enriched model by having each of the n trustees represent one distinct security device. The network topology being symmetric, we can limit our reasoning to one process. Theorem 2 tells us that if any process p is not connected through a trusted path to a majority of trustees, there is no solution to the problem. Since each trustee is behind a distinct process, which is potentially Byzantine, there must be a majority of correct processes. From Theorem 2, we can then say that the guardian angels approach cannot deterministically solve fair exchange, if there is not a majority of correct processes. As one could expect, this result concurs with the result found in [AGGV05].

Nonetheless, a very interesting feature of the approach proposed in [AGGV05] lies in its ability to gracefully degrade its quality of service from deterministic fairness to probabilistic fairness. Figure 1(b) shows an example of the guardian angels topology with five processes.

Fair Exchange in the Pervaho Middleware. In [GR06], we propose a modular algorithm solving fair exchange in the context of the Pervaho middleware [EGH05]. This solution follows the same topology as with guardian angels. This algorithm relies on the use of two key building blocks: a tamperproof secure box module and a module solving the well-known Byzantine agreement problem. The secure boxes are not connected directly with each other and they are only needed in key steps of the algorithm, contrary to the guardian angels approach. Since the network topology is identical to the Guardian angels topology, our enriched model yields the same results in both cases.

5 Concluding Remarks

In this paper, we proposed a formal description of the fair exchange problem which clearly separates liveness and safety, thanks to fine-grained properties. We proved that fair exchange cannot be solved in a synchronous model with Byzantine failures without at least one identified correct process. Based on this result, and by enriching our previous model with identified correct processes (trustees), we defined a generic model for describing a wide range of solutions to the fair exchange problem. We then gave a necessary condition for solving fair exchange in this model. Intuitively, this condition states that each correct process must be reliably connected to a majority of trustees.

Acknowledgements This research is partly funded by the Swiss National Science Foundation, in the context of Project number 200021-104488.

References

[AGGV05] G. Avoine, F. Gärtner, R. Guerraoui, and M. Vukolic. Gracefully Degrading Fair Exchange with Security Modules (Extended Abstract). In *In Proceedings of the 5th European Dependable Computing Conference - EDCC 2005*, 2005.

[ASW00] N. Asokan, V. Shoup, and M. Waidner. Optimistic Fair Exchange of Digital Signatures. *IEEE Journal on Selected Area in Communications*, 18:593–610, 2000.

[Ate99] G. Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In *CCS '99: Proceedings of the 6th ACM conference on Computer and communications security*, pages 138–146, New York, NY, USA, 1999. ACM Press.

[AV03] G. Avoine and S. Vaudenay. Fair Exchange with Guardian Angels. Technical report, Swiss Federal Institute of Technology (EPFL), 2003.

[BP90] H. Bürk and A. Pfitzmann. Value Exchange Systems Enabling Security and Unobservability. *Computers & Security*, 9(9):715–721, 1990.

[BWW00] B. Baum-Waidner and M. Waidner. Round-Optimal and Abuse Free Optimistic Multiparty Contract Signing. In *Automata, Languages and Programming*, number 1853 in Lecture Notes in Computer Science (LNCS), pages 524–535. Springer, 2000.

[CT96] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.

[DGG05] A. Doudou, B. Garbinato, and R. Guerraoui. *Dependable Computing Systems: Paradigms, Performance Issues, and Applications*, chapter Tolerating Arbitrary Failures with State Machine Replication, pages 27–56. Wiley, 2005.

[EGH05] P. Eugster, B. Garbinato, and A. Holzer. Location-based Publish/Subscribe. In *Proceedings of the 4th IEEE International Symposium on Network Computing and Applications (IEEE NCA05)*, Cambridge (MA), July 2005.

[EY80] S. Even and Y. Yacobi. Relations Among Public Key Signature Systems. Technical report, Technion - Israel Institute of Technology, 1980.

[FLP85] M. Fischer, N. Lynch, and M. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *J. ACM*, 32:374–382, April 1985.

[FR97] M.K. Franklin and M.K. Reiter. Fair exchange with a semi-trusted third party (extended abstract). In *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security*, pages 1–5, New York, NY, USA, 1997. ACM Press.

[GR06] B. Garbinato and I. Rickebusch. A Modular Solution to Fair Exchange for Peer-to-Peer Middleware. Technical Report DOP-20060410, University of Lausanne, DOP Lab, 2006.

[HT93] V. Hadzilacos and S. Toueg. Fault-tolerant broadcasts and related problems. pages 97–145, 1993.

[KGM95] S. Ketchpel and H. García-Molina. Making Trust Explicit in Distributed Commerce Transactions. In *Proceedings of the International Conference on Distributed Computing Systems*, 1995.

[MGK02] O. Markowitch, D. Gollmann, and S. Kremer. On Fairness in Exchange Protocols. In *Proceedings of the 5th International Conference Information Security and Cryptology (ICISC 2002)*, volume 2587 of *Lecture Notes in Computer Science*, pages 451–464. Springer, November 2002.

[Mic03] S. Micali. Simple and fast optimistic protocols for fair electronic exchange. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 12–19, New York, NY, USA, 2003. ACM Press.

[PG99] H. Pagnia and F. Gärtner. On the Impossibility of Fair Exchange without a Trusted Third Party. Technical report, Swiss Federal Institute of Technology (EPFL), 1999.

[RRN05] I. Ray, I. Ray, and N. Natarajan. An anonymous and failure resilient fair-exchange e-commerce protocol. *Decision Support Systems*, 39(3):267–292, 2005.

[Sha79] Adi Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.

[SW02] T. Sandholm and X. Wang. (Im)possibility of safe exchange mechanism design. In *Eighteenth national conference on Artificial intelligence*, pages 338–344, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.