An Optimistic NBAC-based Fair Exchange Method for Arbitrary Items

Masayuki Terada, Kensaku Mori, and Sadayuki Hongo

NTT DoCoMo, Inc.,
3–5 Hikari-no-oka, Yokosuka, Kanagawa, Japan

Abstract. Fair exchange protocols are important in realizing safe electronic commerce. In particular, optimistic fair exchange protocols, which involve a trusted third party only when mutual communication between exchanging parties fails, are the most promising development because of their efficiency. Unfortunately, however, existing optimistic protocols place restrictions on the items that can be exchanged, i.e., at least one item must be a "strongly generatable" item such as a digital signature. Without this requirement, only weak fairness that requires (expensive) external dispute resolution processes (e.g. trials in court) after exchange failure can be assured. This paper proposes a novel fair exchange method that enables parties to fairly exchange arbitrary items in an optimistic manner. This is achieved by realizing an optimistic non-blocking atomic commitment (NBAC) protocol between two smartcards and adapting the known result that fair exchange can be reduced to NBAC among trusted processes.

1 Introduction

To enable consumers and merchants to securely participate in electronic commerce without fear of fraud, it is essential to guarantee transaction fairness. When buying digital content such as a movie or music, for example, the content data must be exchanged fairly with the payment data; it must be guaranteed that both data transfers are performed or no valuable data is transferred. Such an exchange of data is called a "fair exchange".

Fair exchange is not easy to achieve. To guarantee fairness, several fair exchange protocols apply gradual data release and others involve a trusted third-party (TTP)[1–3]. In particular, optimistic protocols[4–6] (often called off-line TTP protocols), which involve a TTP in an exchange only when something goes wrong, appear the most promising approach of their efficiency — they require many fewer messages than gradual protocols while greatly reducing TTP overheads.

All existing optimistic protocols place strong restrictions on the data to be exchanged[3]; optimistic protocols can fairly exchange items only when at least one item is a *strong generatable* item, such as a digital signature (or an item which essentially contains a digital signature, e.g. a payment) or both are electronic vouchers[7, 6]. The other exchanges, e.g. the mutual exchange of digital documents, cannot be performed fairly with existing optimistic protocols, and so another (more inefficient) sort of fair exchange protocols such as online TTP protocols must be used for these exchanges. Even if the restriction is satisfied, an exchange may be unfair if the wrong initiator (who starts the exchange) is selected. These limitations of optimistic protocols force users to carefully use different protocols according to the items and situations, and thus make the practical use of fair exchange difficult and inconvenient.

This paper proposes a novel optimistic fair exchange method that fairly exchanges arbitrary items in an optimistic manner with no restriction on items being exchanged. This is achieved by realizing a smartcard-based optimistic[1] non-blocking atomic commitment (NBAC) protocol that solves the NBAC problem in an optimistic manner, along with the application of the known reduction of fair exchange to NBAC among trusted processes[8]. Since neither the protocol nor the reduction algorithm assumes that the exchanged items have any special property, arbitrary items can be fairly exchanged by this method.

The rest of the paper is organized as follows. Section 2 introduces the definition of fair exchange and discusses the characteristics of existing fair exchange protocols, mainly focusing upon the restrictions placed on items by existing optimistic protocols. Section 3 describes the NBAC definition and the reduction algorithm from fair exchange to NBAC, and discusses the practicality of fair exchange based on previous NBAC protocols. Section 4 details the proposed optimistic NBAC protocol. Section 5 analyzes the NBAC properties of the proposed protocol and the feasibility (and the limitations) of the fair exchange method based on this protocol.

2 Fair exchange

Assume that party A and B have items i_{A} and i_{B}, respectively, and A wants to obtain i_{B} from B while B wants i_{A}. Each party has a description of the item to be exchanged; e.g., A knows $d_{\mathrm{B}} = \mathsf{desc}(i_{\mathrm{B}})$.

An exchange protocol that exchanges items i_{A} and i_{B}, between exchanging parties A and B, respectively, is called a fair exchange protocol if it satisfies the following properties when party A behaves correctly (and vice versa)[4][2].

Effectivity If B also behaves correctly[3], and both A and B do not want to abandon the exchange, then when the protocol completed, A has i_{B} such that $d_{\mathrm{B}} = \mathsf{desc}(i_{\mathrm{B}})$.

[1] By *optimistic* we mean that a TTP is involved in an exchange only for exchange resolution. Note that this definition differs from that used in the transaction processing literature, e.g., optimistic transactions, where the resources involved are not locked (to improve concurrency) and any inconsistency is fixed (compensated) later.

[2] In [4], another property called "non-repudiability" is also listed, but we do not discuss this requirement in this paper since it's not a mandatory requirement for fair exchange protocols (as noted in [4])

[3] From A's point of view. That is, failures in the communication channel between A and B are subsumed in the notion of a misbehaving peer.

Termination A can be sure that the protocol will be completed at a certain point in time. At completion, the state of the exchange as of that point is either final or any changes to the state will not degrade the level of fairness achieved by A so far.

Fairness When the protocol has completed, either A has i_{B} such that $d_{\mathrm{B}} = \mathsf{desc}(i_{\mathrm{B}})$, or B has gained no additional information about i_{A}.[4]

Fair exchange protocols can be classified into the following three categories: gradual protocols, online TTP protocols, and optimistic protocols.[1–3]

In gradual protocols, each exchanging party gradually (i.e. in a "bit by bit" manner) releases the item to be exchanged (or the "privilege" to obtain the expected item). They can exchange items without involving any third-party, however, there are drawbacks to practical use because the fairness achieved by this approach is probabilistic and a lot of interactions are needed to achieve fairness with adequate probability.

Online TTP protocols assure (deterministic) fairness by involving a TTP in every exchange[5]. This approach does not require so many messages (usually less than ten) and can assure fairness deterministically. However, involving a TTP in every exchange incurs message congestion at the TTP when many exchanges are concurrently performed, which degrades scalability and the availability of systems implementing the protocol.

Optimistic protocols also utilize a TTP, but only when the mutual interaction cannot be concluded because of a misbehaving partner or failure of the communication channel. This limited use of the TTP resolves the congestion problem of the online TTP protocols while the number of the messages is comparable to (mostly less than) that required by the online TTP protocols; an optimistic protocol typically requires only four messages in errorless cases.[3, 6]

Existing optimistic protocols do, unfortunately, place restrictions on the items to be exchanged and thus cannot fairly exchange arbitrary items; at least one of the items (the item to be received by the initiator of a protocol, actually) must be *strongly generatable*[6][10, 3]. Strong generatability means that a TTP can always generate the item (or its substitution) when the TTP is invoked to conclude the protocol.

A digital signature is a strongly generatable item since a TTP can generate a substitutive signature that has the same effect as the original signature, provided that the exchanging parties agree upon the substitution in advance (namely "the replacement token" method[4]). Electronic vouchers (or electronic rights)[7, 11] can also be assumed to be strongly generatable, but they require

[4] This definition is often called *strong* fairness to distinguish from *weak* fairness which is discussed later.

[5] Protocols wherein the TTP mediates messages are often classified into another category namely "inline TTP protocols"[1, 2].

[6] An optimistic protocol that doesn't require strong generatability but requires a property called "strong revocability" has been proposed[9], but this property is also a strong assumption and only limited items such as closed-loop electronic money are supported.

an exclusive protocol[6] to be fairly exchanged mainly because of the need to prevent duplication of vouchers.

Strong generatability, unfortunately, is not possessed by the many other items, such as digital content and digital documents, exchanged in daily life. Existing optimistic protocols can guarantee fairness when applied to implement *contract signing*, the mutually exchange of two digital signatures, *certified mail*, the exchange of a document and its (signed) receipt, and *voucher trading*, the mutual exchange of two electronic vouchers, however, they guarantee only the weaker notion of fairness called *weak* fairness[7] in other exchanges such as mutual exchange of messages.[3]

Weak fairness requires an external dispute resolution process (e.g. trial in court) if one party wants to recover from some disadvantage. Considering the effort and cost imposed by such a process, weak fairness should be avoided as much as possible.

3 Reducing fair exchange to NBAC

Apart from developing dedicated protocols to solve the fair exchange problem, several researchers are focusing on the similarity between the notion of fair exchange and the problems in distributed computing, e.g., consensus and (non-blocking) atomic commitment, and have analyzed the relationship among these problems[8, 12]. In particular, [8] showed that fair exchange is reducible to NBAC among trusted processes; i.e., the fair exchange problem can be solved if the NBAC problem among processes on smartcards is solved. Since there is no special assumption (such as item generatability) on the items exchanged, a fair exchange achieved by this approach can exchange arbitrary items.

This section introduces the definition of NBAC and the reduction algorithm from fair exchange to NBAC proposed in [8], as well as discussing the practicality of existing NBAC protocols when applied to fair exchange.

3.1 Definition of NBAC

Assume a set of independent processes, each of which has an initial proposed value, yes or no, and try to reach a unanimous decision, commit or abort. A protocol between these processes solves NBAC if it satisfies the following properties[8].

Termination Every correct process eventually reaches a decision.
Agreement No two processes decide differently.

[7] Weak fairness is defined as "if (strong) fairness is not satisfied, then A can prove to an arbiter (a TTP) that B has received (or can still receive) i_{A} such that $\mathsf{desc}(i_{\mathrm{A}}) = d_{\mathrm{A}}$, without any further intervention from A."[4]

[8] C-Validity and A-Validity are often called "non-triviality" and "uniform-validity", respectively.

Commit-Validity (C-Validity) If all processes propose yes and there is no failure, then the decision value must be commit.

Abort-Validity (A-Validity) If at least one process proposes no, then the decision value must be abort.

3.2 Reducing fair exchange to NBAC using smartcards

The following briefly describes how the fair exchange problem can be solved by NBAC (with help of smartcards).

Assume a system that consists of a set of processes interconnected by a communication network with bidirectional synchronous[9] channels. The processes are divided into two classes, *untrusted* processes and *trusted* processes. The trusted processes are assumed to behave correctly (the untrusted processes are not). Each untrusted process is *associated* with a trusted process, and vice versa. The paired untrusted and trusted processes are adjacent; they are directly connected by a communication channel. Any two untrusted processes are adjacent, while no two trusted processes can be adjacent. This system model can be considered to represent the configuration in which hosts (e.g. PCs and mobile phones) are connected to a network and smartcards are connected to hosts (e.g. a smartcard connected to a PC by a card R/W and a SIM card on a mobile phone).

In this system setting, NBAC among the trusted processes solves fair exchange among the untrusted processes by the algorithm described below:

```
**FairExchange**(item $i$, description $d$) {
  ⟨send $i$ to exchange partners over secure channel⟩
  **timed wait for** ⟨expected item $i_e$ from exchange partners⟩
    ⟨check $d$ on $i_e$⟩
  **if** (check succeeds and no timeout)
    **then** $vote$ := yes **else** $vote$ := no **endif**
  $result$ := NBAC($vote$)
  **if** ($result$ = commit)
    **then return** $i_e$ **else return** ⟨abort⟩ **endif**
}
```

The proof that the above algorithm realizes fair exchange is described in [8][10].

3.3 Problems in NBAC-based approach

Although fair exchange can be (rather easily) resolved by applying NBAC as described above, the NBAC problem itself is known as a *hard* problem[13]. Existing NBAC protocols mostly focus much on multi-party settings, and so fail to

[9] While synchronity of channels is assumed in this section as to [8], we will relax this assumption in Sect. 4.

[10] To be accurate, [8] uses a different approach to defining the fair exchange problem (mostly same as the definition in [3]) from the definition described in Sect. 2 (follows the definition in [4], which is more rigorous), but they are basically equivalent in the two-party setting.

match the efficiency of the optimistic protocols introduced in Sect. 2 when used to solve the fair exchange problem.

The well-known two-phase commit (2PC) algorithm that solves the atomic commitment problem (equivalent to NBAC without Termination) cannot solve NBAC, since a crash of the *coordinator* process that gathers votes and distributes a decision may block termination. 2PC can avoid blocking (thus assuring termination) given a resilient coordinator and synchronous channels between the coordinator and other processes; these assumptions might not be infeasible since most TTP-based fair exchange protocols also make similar assumptions (i.e., the TTP are assumed to be eventually reachable from every exchanging party), however, since 2PC requires the coordinator to interact with all processes in every protocol run, the congestion problem is raised as in the online TTP fair exchange protocols.

The three-phase commit (3PC) algorithm, which introduces another phase to elect a coordinator (pre-commit phase), is known to solve NBAC in synchronous systems without assuming a reliable process[14]. However, 3PC is also known as a complicated algorithm that is rarely implemented[15] and introducing an additional phase to 2PC inevitably makes 3PC less efficient than 2PC.

The Monte-Carlo NBAC algorithm[8] solves a weaker variant of NBAC, wherein the Agreement property is satisfied with some probability p $(0 < p < 1)$. This algorithm doesn't involve a coordinator and thus avoids the congestion problem, but probabilistic Agreement implies that achievable fairness is also probabilistic; a large number of interactions are needed to achieve fairness with adequate probability, alike the gradual fair exchange protocols.

4 Optimistic NBAC protocol

As mentioned in Sect. 3, fair exchange based on NBAC among trusted processes doesn't require any special assumptions on the exchanged items. If NBAC can be performed in an optimistic manner, it would be possible to realize optimistic fair exchange that can exchange arbitrary items. This is the key point of our approach.

In this section, we propose an optimistic NBAC protocol between two smartcards. Similar to optimistic fair exchange protocols, this protocol runs in an optimistic manner, i.e., a TTP is involved only when something goes wrong, but solves NBAC rather than fair exchange. This protocol requires only three messages to be passed between the participating smartcards to solve NBAC in errorless cases; this is comparable to existing optimistic protocols, most of which require four messages.

4.1 System model

The system model assumed by this protocol is similar to the system described in Sect. 3.2, except for the following:

- the number of parties (the number of associated pairs) is two,

– a TTP process, which is a reliable trusted process adjacent to each untrusted process is added to the system, and
– untrusted processes are connected by an asynchronous channel instead of a synchronous channel.

The first and second differences are introduced as we consider optimistic exchanges between two parties here. The last difference relaxes the network assumption; because communication channels among hosts pass across a widely-distributed network such as the Internet and mobile phone networks, it is difficult to assume that all communication is synchronous[11].

The channels between each associated pair, i.e., between an untrusted process and a trusted process, stay synchronous as well as those between the TTP and untrusted hosts.

The feasibility of these system assumptions are discussed in Sect. 5.2.

4.2 Protocol

Assume trusted processes A and B (on smartcards), which communicate with each other through associated untrusted processes. Both A and B have their own signing key, kept secret from any other party, and the corresponding public key certificate. $vote_{\mathrm{A}}$ is the input *vote* value of A and $vote_{\mathrm{B}}$ is that of B. $result_{\mathrm{A}}$ and $result_{\mathrm{B}}$ are similar. H() is a collision-resistant one-way hash function (such as SHA1), $\mathrm{Sig}_X(m)$ is m and a digital signature by X's signing key, and $Cert_X$ is X's public key certificate corresponding to X's signing key.

The mutual communication part of this protocol (namely the *main* protocol) is performed as follows. The *abort* subprotocol and the *resolve* subprotocol invoked from this protocol are described later.

1. A generates random number r and calculates $s := H(r)$.
2. A → B: $m_1 := \{\mathrm{Sig}_{\mathrm{A}}(s, vote_{\mathrm{A}}), \mathrm{Cert}_{\mathrm{A}}\}$. If $vote_{\mathrm{A}} = \mathsf{no}$, A terminates the protocol with the output $resultA := \mathsf{abort}$ after sending m_1.
3. B verifies the signature of m_1. If this fails, B waits m_1 again. If the verification succeeds and $vote_{\mathrm{A}} = \mathsf{yes}$, B proceeds to the next step. If $vote_{\mathrm{A}} = \mathsf{no}$ or the reception of (correct) m_1 timeouts, B terminates the protocol with the output $result_{\mathrm{B}} := \mathsf{abort}$.
4. B → A: $m_2 := \{\mathrm{Sig}_{\mathrm{B}}(s, vote_{\mathrm{B}}), \mathrm{Cert}_{\mathrm{B}}\}$.
5. A verifies the signature of m_2. If this fails, A waits m_2 again. If the verification succeeds and $vote_{\mathrm{B}} = \mathsf{yes}$, A proceeds to the next step. If $vote_{\mathrm{B}} = \mathsf{no}$, A terminates the protocol with the output $result_{\mathrm{A}} := \mathsf{abort}$. When the reception of m_2 timeouts, A abandons this protocol (i.e. ignores m_2 even if received later) and invokes the abort subprotocol.

[11] This relaxation does not affect the reducibility from fair exchange to NBAC of the algorithm described in Sect. 3.2, since (besides the NBAC execution) this algorithm use the channels among untrusted processes only to receive item i_e and timing out the reception of i_e.

6. A → B: $m_3 := r$. After sending m_3, A terminates the protocol with the output $result_{\mathrm{A}} := \mathsf{commit}$.
7. Using s in m_1 and r in m_3, B verifies if $s = \mathrm{H}(r)$. If this fails, B waits m_3 again. If the verification succeeds, B terminates the protocol with the output $result_{\mathrm{B}} := \mathsf{commit}$. When the reception of m_3 timeouts, B invokes the resolve subprotocol.

Since the abort subprotocol and the resolve subprotocol are performed in the almost same way, we describe them together. In the following description of the subprotocol(s), P is either A or B that invoked the protocol and $flag$ is a flag that indicates which protocol is being performed; i.e., let $P := \mathrm{A}$ and $flag := \mathsf{abort}$ when performing the abort subprotocol, while $P := \mathrm{B}$ and $flag := \mathsf{commit}$ with the resolve subprotocol.

T is the TTP, which has its own signing key and corresponding certificate $\mathrm{Cert}_{\mathrm{T}}$. T manages two sets S_{abort} and S_{commit}, whose initial states are $S_{\mathsf{abort}} = S_{\mathsf{commit}} = \{\phi\}$.

The abort and resolve subprotocols are performed as follows.

1. P → T: $m_{t1} := \{\mathrm{Sig}_P(flag, s), \mathrm{Cert}_P\}$.
2. T receives m_{t1} and executes the followings:
   (a) verifies the signature of m_{t1}, and waits m_{t1} again if the verification failed, and
   (b) arbitrates whether this protocol should be aborted or committed:
      – if $s \in S_{\mathsf{abort}}$, then let $result_{\mathrm{T}} := \mathsf{abort}$,
      – if $s \in S_{\mathsf{commit}}$, then let $result_{\mathrm{T}} := \mathsf{commit}$, or
      – if neither, then let $S_{flag} := S_{flag} \cup s$ (i.e. add s to S_{abort} when aborting (or S_{commit} when resolving)) and $result_{\mathrm{T}} := flag$.
3. T → P: $m_{t2} := \{\mathrm{Sig}_{\mathrm{T}}(result_{\mathrm{T}}, s), \mathrm{Cert}_{\mathrm{T}}\}$.
4. P verifies the signature of m_{t2}. If this fails, then P waits m_{t2} again. If it succeeds, P terminates the protocol with the output $result_P := result_{\mathrm{T}}$.

5 Discussions

5.1 Analysis of NBAC properties

We discuss below how the protocol proposed in Sect. 4.2 satisfies the NBAC properties in Sect. 3.1. In the following analysis, we assume r has enough length and randomness that the possibility of r being predicted before A reveals m_3 is negligible.

Termination property When there is no failure in the communication channels and either party is honest (the associated untrusted process that forward the messages to the other behaves correctly), both A and B can obviously terminate the protocol with a decision in the main protocol[12].

[12] Since the TTP is not involved in the main protocol, this also confirms that this protocol is an optimistic protocol.

We then discuss the situation in which either process, A or B, can terminate if the execution of the protocol is interupted by failures in the communication channels or a misbehaving partner.

The execution of process A may be interrupted only by the non-arrival of correct m_2. In this case, A timeouts and can determine $result_{\mathrm{A}}$ by invoking the abort subprotocol.

The execution of B may be interrupted by the non-arrival of correct m_1 or m_3. When m_1 doesn't arrive, B can terminate the protocol with $result_{\mathrm{B}} := \mathsf{abort}$ in step 3 of the main protocol. In case m_3 doesn't arrive, B timeouts and can determine $result_{\mathrm{B}}$ by invoking the resolve subprotocol.

Hence, either A or B can terminate the protocol and issue result, and the Termination property is satisfied.

Agreement property First, we show that $(result_{\mathrm{A}} = \mathsf{commit}) \Rightarrow (result_{\mathrm{B}} = \mathsf{commit})$.

Process A may terminate the protocol with $result_{\mathrm{A}} = \mathsf{commit}$ iff it successfully executes step 6 (and terminates) in the main protocol or the TTP arbitrates that the protocol should conclude by commit (i.e. sends m_{t2} where $result_{\mathrm{T}} = \mathsf{commit}$ to A) in the abort subprotocol.

When assuming that A successfully executed step 6 in the main protocol, B must terminate the protocol with $result_{\mathrm{B}} = \mathsf{commit}$ by receiving m_3, or invoke the resolve subprotocol by timeout of m_3, in step 7 of the main protocol. The agreement is obviously reached in the former case. In the latter case, since A is assumed to have successfully executed step 6 in the main protocol, A must not have invoked the abort subprotocol; the resolve subprotocol inevitably concludes with $result_{\mathrm{B}} = \mathsf{commit}$.

If A terminates the protocol by receiving $result_{\mathrm{T}} = \mathsf{commit}$ in the abort subprotocol, T must have received m_{t1} from B in the resolve subprotocol before receiving it from A in the abort subprotocol. In this case, the resolve subprotocol must have concluded (or will conclude) with $result_{\mathrm{B}} = \mathsf{commit}$.

Next, we show the converse, $(result_{\mathrm{B}} = \mathsf{commit}) \Rightarrow (result_{\mathrm{A}} = \mathsf{commit})$.

Process B may terminate the protocol with $result_{\mathrm{B}} = \mathsf{commit}$ iff it successfully received m_3 in step 7 of the main protocol, or it received $result_{\mathrm{T}} = \mathsf{commit}$ in the resolve subprotocol. In the former case, A must have successfully executed step 6 in the main protocol and terminated with $result_{\mathrm{A}} = \mathsf{commit}$. In the latter case, m_{t1} of the abort subprotocol by A must not have arrived at T yet; A has terminated in step 6 in the main protocol, or m_{t1} from A will arrive at T later. In either case, A terminates with $result_A = \mathsf{commit}$.

As a corollary of the above results,

$$(result_{\mathrm{A}} = \mathsf{commit}) \Leftrightarrow (result_{\mathrm{B}} = \mathsf{commit}). \tag{1}$$

Since each process decides either commit or abort, $\neg(result_X = \mathsf{commit}) \Rightarrow (result_X = \mathsf{abort})$, and therefore,

$$(result_{\mathrm{A}} = \mathsf{abort}) \Leftrightarrow (result_{\mathrm{B}} = \mathsf{abort}). \tag{2}$$

Hence, A and B do not decide differently, and the Agreement property is satisfied.

C-Validity property If $(vote_{\mathrm{A}} = \mathsf{yes}) \cap (vote_{\mathrm{B}} = \mathsf{yes})$ and there is no failure, then process A terminates with $result_{\mathrm{A}} = commit$ in step 6 in the main protocol and B terminates with $result_{\mathrm{B}} = commit$ in step 7.

The C-Validity property is satisfied therefore.

A-Validity property If $vote_{\mathrm{A}} = \mathsf{no}$, then process A terminates with $result_{\mathrm{A}} := \mathsf{abort}$ in step 2 in the main protocol. Similarly, B terminates with $result_{\mathrm{B}} := \mathsf{abort}$ in step 3 if $vote_{\mathrm{B}} = \mathsf{no}$. In either case, $result_{\mathrm{A}} = result_{\mathrm{B}} = \mathsf{abort}$ according to Eq. (2).

The A-Validity property is satisfied.

5.2 Feasibility

In the following, we show that this exchange method based on the optimistic NBAC protocol proposed in Sect. 4.2 and the reduction algorithm introduced in Sect. 3.2 can be feasibly implemented under the system assumptions described in Sect. 4.1.

Implementation of the processes Besides communication channels, the system consists of associated pairs of an untrusted process and a trusted process, and a trusted third-party (TTP) process. Each exchange party is assumed to be represented by an associated pair.

An untrusted process is easily implemented on a host connected to a network (e.g. a personal computer and a mobile phone). It is not assumed to be reliable and there are no difficulties to implementing it; what it has to do are to input the item to be sent and a description of the item to be received into the associated trusted process, and to forward the messages among trusted processes.

A trusted process (associated with an untrusted process) is required to be implemented on a tamper-resistant user device such as a smartcard connected to a host through a card reader/writer (card R/W) or embedded in a host (like a SIM card in a mobile phone). If not, an exchange becomes completely unfair because a party can obtain the received item regardless of the result of the NBAC run in the reduction algorithm; tamper-resistance is an essential requirement in applying this method. Assuming such a trusted device might be not so impractical nowadays considering the rapid penetration of smartcards; e.g., most GSM phones and 3G mobile phones have (U)SIM cards. However, regarding implementation on a smartcard, the performance bottlenecks associated with smartcards should be carefully considered. Performance estimation of the smartcard implementation is discussed later.

The TTP process will be implemented as a server connected to a network. This server should be managed by a trusted third-party. This server is involved in an exchange when either of the exchanging parties invokes an abort or resolve subprotocol. Since the protocol is optimistic, these invocations are performed only when something goes wrong in mutual interactions between exchanging parties; when many exchanges are conducted concurrently, the TTP server is involved in a part of them. Under the assumption that failures that need arbitration by the TTP are rare, this system will avoid congestion of messages on the TTP and be scalable, like existing optimistic fair exchange protocols.

Implementation of the channels As described in Sect. 4.1, we assume a system where the channels between untrusted processes are asynchronous and the other channels (the channels between an associated pair and those between the TTP and untrusted processes) are synchronous. When considering to apply the system to exchanges in electronic commerce, the system should be able to be implemented in a widely-distributed and unreliable network such as the Internet and mobile phone networks.

Asynchronous channels between untrusted processes obviously can be implemented in such networks. Since both processes in an associated pair are managed by an exchanging party, the synchronous communication channel between them is mostly local and should be easily implemented; although a user can block interactions between them, e.g., by removing the smartcard from the card R/W, we can treat this as misbehavior of the untrusted process (on the host).

The channels between the TTP process and each untrusted process might not be local; hosts are most likely to communicate with the TTP server through a network. However, different from communicating with another host, which may be unsure where it is and who manages it, the TTP can be expected to be connected to the network continuously. Although the communication channel can be (permanently) lost if the host is disconnected from the network by its user, we can also treat this as misbehavior of an untrusted process. This assumption is, accordingly, also feasible.

Performance The possible bottlenecks of a smartcard implementation are the I/O performance, the processing speed (especially cryptographic calculation) and the number of write operations to non-volatile memory (write operations to EEPROM are much slower than those to RAM). Since no persistent store of data into a smartcard is needed in the proposed exchange, we focus on the I/O interactions and the cryptographic processes.

The proposed NBAC protocol can terminate with unanimous decisions by exchanging three messages between trusted processes in errorless cases (i.e. in the main protocol). In this protocol, the required cryptographic calculations in each trusted process, i.e. each smartcard, are a pair of signature generation and verification (using the corresponding certificate) and a single hash calculation. When something goes wrong in the mutual interaction, an additional round-trip message exchange, which involves another pair of signature generation and verification, with a TTP may be required. This load does not exceed that of existing optimistic protocols, so this protocol can be considered to be practical for implementation on current smartcards.

An optimistic fair exchange protocol for electronic vouchers, which is slightly more complex than the proposed protocol (requires four messages and almost the same cryptographic calculations), performs an exchange in less than two seconds when implemented on mid-range smartcards.[16] This implies that the proposed NBAC protocol can be performed in the same or less time.

On the other hand, the reduction algorithm seems to need some performance improvement for applications that exchange huge data. This algorithm requires trusted processes to send and receive the whole items to be exchanged. However, since the I/O performance of the current smartcards is not so high (approximately 10kbps $\sim$ 400kbps), it might be infeasible to exchange a large amount of data, such as digital content. Some measure, e.g., exchanging encrypted items in advance and exchanging the description keys fairly, could be introduced to exchange large items, but it may make the item verification (i.e. checking the description of the expected item on the received item) difficult. This could be an open problem.

5.3 Limitation

Although the proposed NBAC protocol and fair exchange method is more efficient than existing NBAC protocols and fair exchange protocols that can exchange arbitrary items (i.e., gradual and online TTP protocols), respectively, it cannot replace either protocols in general; it lacks applicability to multi-party settings and relies upon trusted devices and a TTP.

Since most exchanges are conducted by two parties, e.g., trades in commerce, the support for two-party setting should be attractive enough to stimulate the frequency of fair exchange. However, distributed transactions often need consistency in three or more (independently managed) resources. This protocol cannot be applied to guarantee atomic commit in such transactions. Secure multi-party computation, which often requires fairness, is another example. These applications will need another protocol to assure consistency or fairness.

Relying upon trusted devices and a TTP should not spoil the feasibility as discussed in Sect. 5.2, however, there may be environments where assuming them is difficult. Our method cannot be adopted in such environments.

5.4 Comparison to existing smartcard-based exchange protocols

Several protocols have been proposed for fair exchange via smartcards.

One of them is a reduction from fair exchange to a weaker variant of NBAC (Monte-Carlo NBAC)[8], mentioned in Sect. 3. As described before, this has similar characteristic to gradual exchange protocols and shares the same problems (probabilistic fairness and high interaction costs).

A similar approach based on (a variant of) the consensus problem (named Biased Consensus) among trusted processes is introduced in [12]. This achieves deterministic fair exchange if a majority of the untrusted processes are honest (behave correctly), but fairness becomes probabilistic if half or more processes are dishonest; in two-party exchanges, which will be the most common setting, a misbehaved partner is enough to lose the honest majority and thus only probabilistic fairness can be assured.

In [17, 18], several protocols that use a smartcard to achieve optimistic fair exchange between two parties are introduced. These protocols exchange items between a party called *vendor* and another party *customer*, who uses a smartcard (the vendor does not use a smartcard). One of the protocols, called the basic protocol, can exchange arbitrary items[13]; it does not assume generatability or revocability on either item. However, these protocols have a drawback in that the Termination property (of fair exchange) is not assured to the vendor; the vendor is not assured to be able to know if its item is sold or not.

Another smartcard-based optimistic fair exchange protocol is proposed in [6]. This protocol fairly exchanges electronic vouchers stored in smartcards as well as preventing illegal acts on the vouchers (forgery, alteration, and duplication). This protocol can be applied only to the exchange of particular items, i.e., vouchers stored in smartcards, as it is[14].

6 Conclusion

In this paper, we argued that existing optimistic protocols place an excessive restriction on exchanged items, i.e., at least one of the item has to be a strongly generatable item such as a digital signature. To circumvent this restriction, we focused on NBAC-based fair exchange using smartcards, which is based on NBAC between trusted processes, and proposed a novel NBAC protocol that can be performed between smartcards in an optimistic manner. The NBAC properties and feasibility of the protocol were also discussed; the proposed protocol satisfies all of the NBAC requirements and can be efficiently implemented on current smartcards.

NBAC does not only realize fair exchange but is also useful for diverse transactions in electronic commerce. Considering the popularity of mobile phones equipped with smartcards and the rapid improvement of smartcard process technologies, this protocol may broadly contribute to realize safe and secure electronic commerce.

[13] The other protocols require one of the items (from the customer to the vendor) to be strongly revocable.

[14] However, interestingly, its use of a different notion of fairness to prevent duplication of vouchers assures that if one party successfully terminates (i.e. commit) an exchange, then the other party *never* aborts the exchange; this implies the Agreement property of NBAC, while the (original) fairness property doesn't (a misbehaved party can abort if the other party successfully commits). Actually, NBAC is reducible to fair voucher exchange (cf. fair exchange is reducible to NBAC, but not vice versa) and the proposed NBAC protocol can also be considered as a generalization of the voucher exchange protocol.

References

1. Zhou, J., ed.: Non-repudiation in electronic commerce. Artech House, Norwood, MA, USA (2001)
2. Kremer, S., Markowitch, O., Zhou, J.: An intensive survey of fair non-repudiation protocols. Computer Communications **25** (2002) 1606–1621
3. Pagnia, H., Vogt, H., C.Gärtner, F.: Fair exchange. The Computer Journal **46** (2003) 55–75
4. Asokan, N.: Fairness in Electronic Commerce. PhD thesis, University of Waterloo (1998)
5. Schunter, M.: Optimistic Fair Exchange. PhD thesis, Universität des Saarlandes (2000)
6. Terada, M., Iguchi, M., Hanadate, M., Fujimura, K.: An optimistic fair exchange protocol for trading electronic rights. In: Proc. 6th Working Conference on Smart Card Research and Advanced Applications (CARDIS'04), IFIP (2004) 255–270
7. Fujimura, K., Eastlake, D.: RFC 3506: Requirements and Design for Voucher Trading System (VTS). (2003)
8. Avoine, G., Gärtner, F., Guerraoui, R., Kursawe, K., Vaudenay, S., Vukolic, M.: Reducing fair exchange to atomic commit. Technical Report 200411, Swiss Federal Institute of Technology (EPFL), School of Computer and Communication Sciences, Lausanne, Switzerland (2004)
9. Vogt, H.: Asynchronous optimistic fair exchange based on revocable items. In: Proc. 7th International Financial Cryptography Conference, IFCA (2003) 208–222
10. Vogt, H., Pagnia, H., Gärtner, F.C.: Modular fair exchange protocols for electronic commerce. In: Proc. 15th Annual Computer Security Applications Conference. (1999) 3–11
11. Terada, M., Kuno, H., Hanadate, M., Fujimura, K.: Copy prevention scheme for rights trading infrastructure. In: Proc. 4th Working Conference on Smart Card Research and Advanced Applications (CARDIS'00), IFIP (2000) 51–70
12. Avoine, G., Gärtner, F., Guerraoui, R., Vukolic, M.: Gracefully degrading fair exchange with security modules. In: Proc. 5th European Dependable Computing Conference (EDCC). (2005) 55–71
13. Guerraoui, R.: Revisiting the relationship between non-blocking atomic commitment and consensus. In: Proc. 9th International Workshop on Distributed Algorithms (WDAG95). (1995) 87–100
14. Skeen, D.: Nonblocking commit protocols. In: Proc. 1981 ACM SIGMOD International Conference on Management of Data. (1981) 133–142
15. Gray, J., Lamport, L.: Consensus on transaction commit. Technical Report MSR-TR-2003-96, Microsoft Research (2004)
16. Terada, M., Mori, K., Ishii, K., Hongo, S., Usaka, T., Koshizuka, N., Sakamura, K.: TENeT: A framework for distributed smartcards. In: Proc. 2nd International Conference on Security in Pervasive computing (SPC2005). Volume 3450 of LNCS., Springer-Verlag (2005) 3–17
17. Vogt, H., Pagnia, H., Gärtner, F.C.: Using smart cards for fair exchange. In: Proc. 2nd International Workshop on Electronic Commerce (WELCOM 2001). (2001) 101–113
18. Vogt, H., Gärtner, F.C., Pagnia, H.: Supporting fair exchange in mobile environments. ACM/Kluwer Journal on Mobile Network and Applications (MONET) **8** (2003) 127–136