Rational Exchange – A Formal Model Based on Game Theory*

Levente Buttyán and Jean-Pierre Hubaux

Institute for Computer Communications and Applications
Swiss Federal Institute of Technology – Lausanne
EPFL-DSC-ICA, CH-1015 Lausanne, Switzerland

Abstract. We introduce game theory as a formal framework in which exchange protocols can be modeled and their properties can be studied. We use this framework to give a formal definition for rational exchange relating it to the concept of Nash equilibrium in games. In addition, we study the relationship between rational exchange and fair exchange. We prove that fair exchange implies rational exchange, but the reverse is not true. The practical consequence of this is that rational exchange protocols may provide interesting solutions to the exchange problem by representing a trade-off between complexity and what they achieve. They could be particularly useful in mobile e-commerce applications.

1 Introduction

There are many applications where two parties have to exchange their digital items via communication networks. Examples include electronic contract signing, certified e-mail, and purchase of network delivered services. An inherent problem in these applications is that a misbehaving party may bring the other party in a disadvantageous situation. For instance, a service provider may deny service provision after receiving payment from a user. This may discourage the parties and hinder otherwise desired transactions.

There are two different approaches proposed in the literature to solve this problem that we can characterize as follows:

- **Fair exchange.** A fair exchange protocol guarantees for a correctly behaving party (i.e., a party that follows the protocol) that it cannot suffer any disadvantages – no matter whether the other party behaves correctly or tries to cheat. Thus, executing the protocol faithfully is safe for both parties.
- **Rational exchange.** A rational exchange protocol provides incentives so that rational (self-interested) parties have more reason to follow the protocol faithfully than to deviate from it. In other words, if one of the parties deviates from the protocol, then she may bring the other, correctly behaving party in a disadvantageous situation, but she cannot gain any advantages by the

misbehavior. Therefore, cheating is uninteresting, and should happen only rarely.

Fair exchange has been extensively studied by the research community (see e.g., [8] for a comprehensive overview). On the other hand, rational exchange has received less attention. We are aware of a few protocols [4, 10, 2] that seem to be rational; however, it is difficult to tell what exactly these protocols achieve, because no formal definition of rational exchange exists. In addition, there seems to be some confusion regarding the relationship between fair exchange and rational exchange. The authors of [4] and [10], for instance, refer to fair exchange in the title of their papers, although the protocols that they describe are rational exchange protocols.

In order to better understand what rational exchange is, in this paper, we give a formal definition for it. Our approach is based on game theory. First, we introduce a framework to model an exchange protocol as a game. Then, we define the concept of rational exchange in terms of a Nash equilibrium in this game. Finally, we study the relationship between rational exchange and fair exchange, and show that fair exchange implies rational exchange, but the reverse is not true in general.

Our theoretical results, thus, justify the intuition that a fair exchange protocol achieves stronger guarantees than a rational exchange protocol. However, the "price" of this is that a fair exchange protocol is usually more complex than a rational exchange protocol. Fair exchange protocols proposed in the literature require either a lot of computation or communication, or a trusted third party that assists the exchange. Rational exchange protocols, on the other hand, may not use any trusted third party, while still keeping computational and communication complexity at a low level [10], or they may take advantage of an already existing trusted party in the system, without requiring additional functions from it [4, 2].

An interesting application area of rational exchange could be mobile e-commerce. In mobile systems, devices are often battery powered and bandwidth is a scarce resource. Therefore, protocols that require a lot of computation or communication are not desirable. In addition, connectivity is sporadic, which means that a trusted third party that resides in the fixed infrastructure may often be unavailable. Thus, fair exchange protocols may not be practical for certain mobile e-commerce applications. On the other hand, rational exchange protocols may provide interesting solutions in these applications by representing a trade-off between complexity and what they achieve.

1.1 An example for rational exchange

As an example, let us consider the following payment protocol:

$U \to V:\ m_1 = (U, V, tid, val, h(rnd), \sigma_U(U, V, tid, val, h(rnd)))$
$V \to U:\ m_2 = srv$
$U \to V:\ m_3 = rnd$
if V received m_1 and m_3:
  $V \to B:\ m_4 = (m_1, rnd, \sigma_V(m_1, rnd))$
if V received only m_1:
  $V \to B:\ m_4' = (m_1, \sigma_V(m_1))$

The above protocol can be used for transferring payment from a user U to a vendor V in exchange for some service provided by V to U. Besides the main parties U and V, the protocol uses a trusted third party, the bank B. We assume that before starting the protocol, U and V have already agreed on the details of the transaction. In particular, we assume that U and V agreed on the value *val* of the payment that U is supposed to pay to V, and the description of the service *srv* that V is supposed to provide to U. In addition, we assume that U and V also agreed on a fresh transaction identifier *tid*.

U starts the protocol by generating a random number *rnd* and computing its hash value $h(rnd)$. Then, she generates the digital signature $\sigma_U(U, V, tid, val, h(rnd))$, and sends m_1 to V. When V receives m_1, it provides the service to U (represented by sending $m_2 = srv$). If U is satisfied, then she reveals the random number *rnd* to V. If V received m_1 and m_3, then it generates the digital signature $\sigma_V(m_1, rnd)$, and sends m_4 to B. If V received only m_1, then it generates the digital signature $\sigma_V(m_1)$, and sends m_4' to B.

Upon reception of m_4, B verifies that it has never processed a transaction between U and V with the transaction identifier *tid* before by looking up its internal database, where it logs all processed transactions. Then, it verifies that the hash value of *rnd* equals the hash value in m_1, and the digital signatures of U and V in the message are valid. If these verifications are successful, then it logs the transaction, and transfers the value *val* from the account of U to the account of V. Upon reception of m_4', B performs similar verifications, and if these are successful, then B logs the transaction, and it debits U's account with the value *val*, but it does *not* credit V's account.

What does this protocol achieve? It is clear that it does not provide fairness, since any of the parties can bring the other, correctly behaving party in a disadvantageous situation. For instance, U can refuse to reveal *rnd* to V. In this case, V can send only m_4' to B, which means that V does not get paid for the service that it provided. Similarly, V can refuse the provision of the service after having received m_1 from U. In this case, V can send m_4' to B, which means that U's account is debited, although she did not receive any service.

On the other hand, note that none of the parties gain any (financial) advantages by cheating. The reason is that V cannot obtain any money without providing the service to U (since U reveals *rnd* only if she received the service); and U cannot receive any service without being charged (since V provides the service only if it received m_1, in which case it can send at least m_4' to B). This means that none of the parties is interested in deviating from the protocol. In a word, the protocol seems to be a rational exchange protocol. Indeed, this can be proven formally within the model that we will introduce in this paper.

2 Preliminaries

In this section, we introduce some basic definitions from game theory [7] that we will use later.

2.1 Extensive games

An *extensive game* is a tuple $\langle P, Q, p, (\mathcal{I}_i)_{i \in P}, (\leq_i)_{i \in P} \rangle$, where

- P is a set of *players*;
- Q is a set of *action sequences* that satisfies the following properties:
  - the empty sequence ϵ is a member of Q,
  - if $(a_k)_{k=1}^{w} \in Q$ and $0 < v < w$, then $(a_k)_{k=1}^{v} \in Q$,
  - if an infinite action sequence $(a_k)_{k=1}^{\infty}$ satisfies $(a_k)_{k=1}^{v} \in Q$ for every positive integer v, then $(a_k)_{k=1}^{\infty} \in Q$;

  If q is a finite action sequence and a is an action, then $q.a$ denotes the finite action sequence that consists of q followed by a. An action sequence $q \in Q$ is *terminal* if it is infinite or if there is no a such that $q.a \in Q$. The set of terminal action sequences is denoted by Z. For every non-terminal action sequence $q \in Q \setminus Z$, $A(q)$ denotes the set $\{a : q.a \in Q\}$ of *available actions* after q.
- p is a *player function* that assigns a player in P to every action sequence in $Q \setminus Z$;
- $\mathcal{I}_i$ is an *information partition* of player $i \in P$, which is a partition of the set $\{q \in Q \setminus Z : p(q) = i\}$ with the property that $A(q) = A(q')$ whenever q and q' are in the same *information set* $I_i \in \mathcal{I}_i$;
- $\leq_i$ is a preference relation of player $i \in P$ on Z.

The interpretation of an extensive game is the following: Each action sequence in Q represents a possible history of the game. The action sequences that belong to the same information set $I_i \in \mathcal{I}_i$ are indistinguishable to player i. This means that i knows that the history of the game is an action sequence in I_i but she does not know which one. The empty sequence ϵ represents the starting point of the game. After any non-terminal action sequence $q \in Q \setminus Z$, player $p(q)$ chooses an action a from the set $A(q)$. Then q is extended with a, and the history of the game becomes $q.a$. The action sequences in Z represent the possible outcomes of the game. If $q, q' \in Z$ and $q \leq_i q'$, then player i prefers the outcome q' to the outcome q.

The preference relations of the players are often represented in terms of *payoffs*: a vector $y(q) = (y_i(q))_{i \in P}$ of real numbers is assigned to every terminal action sequence $q \in Z$ in such a way that for any $q, q' \in Z$ and $i \in P$, $q \leq_i q'$ iff $y_i(q) \leq y_i(q')$.

Conceptually, an extensive game can be thought of as a tree. The edges and the vertices of the tree correspond to actions and action sequences, respectively. A distinguished vertex, called the root, represents the empty sequence ϵ. Every other vertex u represents the sequence of the actions that belong to the edges of the path between the root and u. Let us call a vertex u terminal if the path between the root and u cannot be extended beyond u. Terminal vertices represent the terminal action sequences in the game. Each non-terminal vertex u is labeled by $p(q)$ where $q \in Q \setminus Z$ is the action sequence that belongs to u. Finally, the terminal vertices may be labeled with payoff vectors to represent the preference relations of the players.

2.2 Strategy

A *strategy of player* $i \in P$ is a function s_i that assigns an action in $A(q)$ to every non-terminal action sequence $q \in Q \setminus Z$ such that $p(q) = i$ and q is *consistent* with the strategy, with the restriction that it assigns the same action to q and q' whenever q and q' are in the same information set of i. A non-terminal action sequence $(a_k)_{k=1}^{w} \in Q \setminus Z$ is consistent with the strategy s_i of player i iff $s_i((a_k)_{k=1}^{v}) = a_{v+1}$ for every $0 \leq v < w$ such that $p((a_k)_{k=1}^{v}) = i$. We denote the set of all strategies of player i by S_i.

A *strategy profile* is a vector $(s_i)_{i \in P}$ of strategies, where each s_i is a member of S_i. Sometimes, we will write $(s_j, (s_i)_{i \in P \setminus \{j\}})$ instead of $(s_i)_{i \in P}$ in order to emphasize that the strategy profile specifies strategy s_j for player j.

2.3 Nash equilibrium

Let $o((s_i)_{i \in P})$ denote the resulting outcome when the players follow the strategies in the strategy profile $(s_i)_{i \in P}$. In other words, $o((s_i)_{i \in P})$ is the (possibly infinite) action sequence $(a_k)_{k=1}^{w} \in Z$ such that for every $0 \leq v < w$ we have that $s_{p((a_k)_{k=1}^{v})}((a_k)_{k=1}^{v}) = a_{v+1}$. A strategy profile $(s_i^*)_{i \in P}$ is a *Nash equilibrium* iff for every player $j \in P$ and every strategy $s_j \in S_j$ we have that

$$o(s_j, (s_i^*)_{i \in P \setminus \{j\}}) \leq_j o(s_j^*, (s_i^*)_{i \in P \setminus \{j\}})$$

This means that if every player i other than j follows s_i^*, then player j is also motivated to follow s_j^*, because she loses or at least does not gain anything by not doing so.

3 Protocol games

Let us consider a set of principals that want to execute a given exchange protocol. Moreover, let us assume that the principals may misbehave, and may not follow the protocol faithfully. In such a situation,

- each principal has choices at various stages during the interaction with the others (e.g., to quit the protocol or to continue);

– the decisions that the principals make determine the outcome of their interaction;
– in order to achieve the most preferable outcome, a principal may follow a plan that does not coincide with the faithful execution of the exchange protocol.

Considering its striking similarity to games, it appears to be a natural idea to model this situation with a game. We refer to this game as the *protocol game*.

We want that the protocol game of an exchange protocol models all the possible ways in which the protocol participants can misbehave *within the context of the protocol*. The crucial point here is to make the difference between misbehavior within the context of the protocol and misbehavior in general. Letting the protocol participants misbehave in any way they can would lead to a game that would allow interactions that has nothing to do with the protocol being studied. Clearly, such a game would not be a good model of the protocol, because it would be far too rich, and we suspect that it would be difficult to analyze, and thus, to draw interesting conclusions about the protocol. Therefore, we want to limit the possible misbehavior of the protocol participants. However, we must do so in such a way that we do not lose generality. Essentially, the limitation that we will impose on protocol participants is that they can send only messages that are *compatible* with the protocol. We make this precise in the following paragraphs.

We consider an exchange protocol to be a description $\pi(L)$ of a distributed computation on a set L of parameters. $\pi(L)$ consists of a set $\{\pi_1(L_1), \pi_2(L_2), \ \ldots\}$ of descriptions of local computations, where each L_k is a subset of L. We call these descriptions of local computations shortly *programs*. Each program $\pi_k(L_k)$ is meant to be executed by a protocol participant. Typically, each $\pi_k(L_k)$ contains instructions to wait for messages that satisfy certain conditions. When such an instruction is reached, the local computation can proceed only if a message that satisfies the required condition is provided (or a timeout occurs). Let us call a message m *compatible* with $\pi_k(L_k)$ if the local computation described by $\pi_k(L_k)$ can reach a state in which a message is expected and m would be accepted. Let us denote the set of messages that are compatible with $\pi_k(L_k)$ by $M_{\pi_k}(L_k)$. Then, the set of messages that are compatible with the protocol is defined as $M_\pi(L) = \cup_k M_{\pi_k}(L_k)$.

Apart from requiring the protocol participants to send messages that are compatible with the protocol, we do not impose further limitations on their behavior. In particular, we allow the protocol participants to quit the protocol at any time, or to wait for some time without any activity. Furthermore, the protocol participants can send any messages (compatible with the protocol) that they are able to compute in a given state.

We assume that the network that is used by the protocol participants to communicate with each other is reliable, which means that it always delivers messages to their intended destinations within a constant time interval. Such a network allows the protocol participants to run the protocol in a synchronous fashion. We will model this by assuming that the protocol participants interact with each other in synchronous *rounds*, where each round consists of the following two phases:

1. each participant generates some messages based on her current state, and sends them to some other participants;
2. each participant receives the messages that were sent to her in the current round, and performs a state transition based on her current state and the received messages.

We adopted this approach from [6], where the same model is used to study the properties of distributed algorithms in a synchronous network system. In [3], we also introduce a more general asynchronous model, where we assume that the network is unreliable. Due to space limitation, we do not describe that model here.

3.1 Players

We model each protocol participant (i.e., the two main parties and the trusted third party if there is any) as a player. In addition, we model the communication network as a player too. Therefore, the player set P of the protocol game is defined as $P = \{p_1, p_2, p_3, net\}$, where p_1 and p_2 represent the two main parties of the protocol, p_3 stands for the trusted third party, and net denotes the network. If the protocol does not use a trusted third party, then p_3 is omitted. We denote the set $P \setminus \{net\}$ by P'.

It might seem that it is useless to model the trusted third party explicitly as a player, because it always behaves correctly, and thus, its actions are fully predictable. Nevertheless, we model the trusted third party in the same way as we model the main parties, which leads to a more uniform model. We will make the distinction between the trusted third party and the potentially misbehaving main parties in another way: We restrict the player that represents the trusted third party to follow a particular strategy, whereas we allow the players that represent the potentially misbehaving main parties to choose among several strategies.

3.2 Information sets

Each player $i \in P$ has a local state $\Sigma_i(q)$ that represents all the information that i has obtained after the action sequence q. If for two action sequences q and q', $\Sigma_i(q) = \Sigma_i(q')$, then q and q' are indistinguishable to i. Therefore, two action sequences q and q' belong to the same information set of i iff it is i's turn to move after both q and q', and $\Sigma_i(q) = \Sigma_i(q')$.

We define two types of events: send and receive events. The send event $snd(m, j)$ is generated for player $i \in P'$ when she submits a message $m \in M_\pi(L)$ with intended destination $j \in P'$ to the network, and the receive event $rcv(m)$ is generated for player $i \in P'$ when the network delivers a message $m \in M_\pi(L)$ to i. We denote the set of all events by E.

The local state $\Sigma_i(q)$ of player $i \in P'$ after action sequence q is defined as a tuple $\langle \varrho_i(q), \alpha_i(q), H_i(q) \rangle$, where

- $\varrho_i(q) \in N$ is a positive integer that represents the round number for player i after action sequence q;

- $\alpha_i(q) \in \{\mathsf{true}, \mathsf{false}\}$ is a boolean, which is true iff player i is still active (i.e., she has not quitted the protocol) after action sequence q;
- $H_i(q) \subseteq E \times N$ is player i's local history after action sequence q, which contains the events that were generated for i together with the round number of their generation.

Initially, $\varrho_i(\epsilon) = 1$, $\alpha_i(\epsilon) = \mathsf{true}$, and $H_i(\epsilon) = \emptyset$ for every player $i \in P'$.

The local state $\Sigma_{net}(q)$ of the network consists of a set $M_{net}(q) \subseteq M_\pi(L) \times P'$, which contains those messages together with their intended destination that were submitted to the network in the current round and have not been delivered yet. We call the set $M_{net}(q)$ the network buffer. Initially, $M_{net}(\epsilon) = \emptyset$.

3.3 Available actions

In order to determine the set of actions available for a player $i \in P'$ after an action sequence q, we first tag each message $m \in M_\pi(L)$ with a vector $(\phi_i^m(\Sigma_i(q)))_{i \in P'}$ of conditions. Each $\phi_i^m(\Sigma_i(q))$ is a logical formula that describes the condition that must be satisfied by the local state $\Sigma_i(q)$ of player i in order for i to be able to send message m after action sequence q. Our intention is to use these conditions to capture the assumptions about cryptographic primitives at an abstract level. For instance, it is often assumed that a valid digital signature $\sigma_i(m)$ of player i on message m can only be generated by i. This means that a message $m' \in M_\pi(L)$ that contains $\sigma_i(m)$ can be sent by a player $j \neq i$ iff j received a message that contained $\sigma_i(m)$ earlier. This condition can be expressed by an appropriate logical formula for every $j \neq i$.

Now, let us consider an action sequence q, after which player $i \in P'$ has to move. There is a special action, called quit_i, which is always available for i after q. Every other action available for i after q is a subset of the set $B_i(\Sigma_i(q))$ of *action atoms*, which is defined as

$$B_i(\Sigma_i(q)) = \{\mathsf{send}_i(m, j) : m \in M_\pi(L),\ \phi_i^m(\Sigma_i(q)) = \mathsf{true},\ j \in P' \setminus \{i\}\}$$

The set $A_i(\Sigma_i(q))$ of available actions of player $i \in P'$ after action sequence q is then defined as

$$A_i(\Sigma_i(q)) = \{\mathsf{quit}_i\} \cup \mathcal{B}_i(\Sigma_i(q))$$

where $\mathcal{B}_i(\Sigma_i(q))$ denotes the power set of $B_i(\Sigma_i(q))$.

The interpretation of the above formulae is the following: every player other than the network can quit the protocol at any time, or if she does not quit, then she can send any subset of the messages that she is able to send in her current local state to any other player in P'. Note that $\emptyset \in A_i(\Sigma_i(q))$. By convention, choosing the action $\emptyset$ means that the player does nothing.

Let us consider now an action sequence q, after which the network has to move. There is only one action, called $\mathsf{deliver}(M_{net}(q))$, that is available for the network after q. Thus, $A_{net}(\Sigma_{net}(q)) = \{\mathsf{deliver}(M_{net}(q))\}$. The interpretation of this is straightforward: the only available action of the network is to deliver all messages in the network buffer to their intended destinations.

The above defined actions change the local states of the players as follows:

- If a player $i \in P'$ performs the action quit_i, then the activity flag of i is set to false. The state of every other player $j \in P \setminus \{i\}$ remains the same as before.
- If a player $i \in P'$ performs the action $\emptyset$, then the state of every player $j \in P$ remains the same as before.
- If a player $i \in P'$ performs an action W such that $W \neq \mathsf{quit}_i$ and $W \neq \emptyset$, then the messages in W, together with their intended destination, are inserted in the network buffer, and the corresponding send events are generated for i. The state of every other player $j \in P \setminus \{i, net\}$ remains the same as before.
- If the network performs the action $\mathsf{deliver}(M_{net}(q))$, then for every message-destination pair $(m, j) \in M_{net}(q)$, a $rcv(m)$ event is generated for j if j is still active. Then, every message is removed from the network buffer, and the round number of every active player is increased by one.

3.4 Action sequences and player function

The game is played in repeated rounds, where each round consists of the following two phases: (1) each active player in P' moves, one after the other, in order; (2) the network moves. The game is finished when every player in P' becomes inactive.

In order to make this precise, let us denote the set of players that are still active after action sequence q and have an index larger than v by $P'(q, v)$. Formally, $P'(q, v) = \{p_k \in P' : \alpha_{p_k}(q) = \mathsf{true},\ k > v\}$. Furthermore, let $k_{min}(q, v)$ denote the smallest index in $P'(q, v)$, which is defined as $k_{min}(q, v) = \min_{k: p_k \in P'(q,v)}\ k$.

We define the set Q of action sequences and the player function p of the protocol game together in an inductive manner. By definition, $\epsilon \in Q$. Moreover, $p(\epsilon) = p_1$. In addition,

- if an action sequence q is in Q and $p(q) = p_v$, then
  - $q.a \in Q$ for every $a \in A_{p_v}(\Sigma_{p_v}(q))$;
  - if $P'(q.a, v) \neq \emptyset$, then $p(q.a) = p_{k_{min}(q.a,v)}$, otherwise $p(q.a) = net$;
- if an action sequence q is in Q and $p(q) = net$, then
  - $q.a \in Q$ for the single action $a = \mathsf{deliver}(M_{net}(q)) \in A_{net}(\Sigma_{net}(q))$;
  - if $P'(q.a, 0) \neq \emptyset$, then $p(q.a) = p_{k_{min}(q.a,0)}$, otherwise $q.a$ is a terminal action sequence.

3.5 Payoffs

Let us denote the items that the main parties of the protocol represented by p_1 and p_2 want to exchange in the protocol by γ_{p_1} and γ_{p_2}, respectively. Typically, γ_{p_1} and γ_{p_2} are members of the parameter set L of the protocol. Furthermore, let us denote the value that γ_j is worth to i $(i, j \in \{p_1, p_2\})$ by $u_i(\gamma_j)$. In practice, it may be difficult to quantify $u_i(\gamma_j)$. However, our approach does not depend on the exact value of $u_i(\gamma_j)$; we require only that $u_{p_1}(\gamma_{p_2}) > u_{p_1}(\gamma_{p_1}) > 0$ and $u_{p_2}(\gamma_{p_1}) > u_{p_2}(\gamma_{p_2}) > 0$ hold. We consider these to be necessary conditions for the exchange to take place at all.

The payoff $y_i(q)$ for player $i \in \{p_1, p_2\}$ assigned to the terminal action sequence q is defined as $y_i(q) = y_i^+(q) - y_i^-(q)$. We call $y_i^+(q)$ the *income* and $y_i^-(q)$ the *expenses* of i, and define them as follows:

$$y_i^+(q) = \begin{cases} u_i(\gamma_j) & \text{if } \phi_i^+(q) = \mathsf{true} \\ 0 & \text{otherwise} \end{cases}$$

$$y_i^-(q) = \begin{cases} u_i(\gamma_i) & \text{if } \phi_i^-(q) = \mathsf{true} \\ 0 & \text{otherwise} \end{cases}$$

where $j \in \{p_1, p_2\}$, $j \neq i$, and $\phi_i^+(q)$ and $\phi_i^-(q)$ are logical formulae. The exact form of $\phi_i^+(q)$ and $\phi_i^-(q)$ depends on the particular exchange protocol being modeled, but the idea is that $\phi_i^+(q) = \mathsf{true}$ iff i gains access to γ_j, and $\phi_i^-(q) = \mathsf{true}$ iff i loses control over γ_i in q. A typical example would be $\phi_i^+(q) = (\exists \varrho : (rcv(m), \varrho) \in H_i(q))$, where we assume that m is the only message in $M_\pi(L)$ that contains γ_j.

Note that according to the above definition, the payoff $y_i(q)$ of player i can take only four possible values: 0, $u_i(\gamma_j)$, $-u_i(\gamma_i)$, and $u_i(\gamma_j) - u_i(\gamma_i)$ for every terminal action sequence q of the protocol game.

Since we are only interested in the payoffs of p_1 and p_2 (i.e., the players that represent the main parties), we define the payoff of every other player in $P \setminus \{p_1, p_2\}$ to be 0 for every terminal action sequence of the protocol game.

Before leaving this section, we define a property of exchange protocols:

Definition 1. *Let us consider a two-party exchange protocol $\pi(L)$ and its protocol game $G_\pi(L)$. $\pi(L)$ is said to be* closed for incomes *iff for every terminal action sequence q of $G_\pi(L)$ we have that*

$$y_{p_1}^+(q) > 0 \Rightarrow \; y_{p_2}^-(q) > 0$$
$$y_{p_2}^+(q) > 0 \Rightarrow \; y_{p_1}^-(q) > 0$$

If a protocol is closed for incomes, then no income comes from outside of the system. This means that if a main party gains something, then the other main party must lose something. Exchange protocols usually satisfy this property. On the other hand, exchange protocols are not necessarily closed for expenses: a main party may lose something without the other main party gaining anything.

4 Formal definition of rational exchange

Informally, a two-party rational exchange protocol is an exchange protocol in which both main parties are motivated to behave correctly and to follow the protocol faithfully. If one of the parties deviates from the protocol, then she may bring the other, correctly behaving party in a disadvantageous situation, but she cannot gain any advantages by the misbehavior. This is very similar to the concept of Nash equilibrium in games. This inspired us to give a formal definition of rational exchange in terms of a Nash equilibrium in the protocol game.

Let us consider an exchange protocol $\pi(L) = \{\pi_1(L_1), \pi_2(L_2), \ \ldots\}$ and its protocol game $G_\pi(L)$ constructed according to the framework described in the previous section. Recall that each $\pi_k(L_k)$ is a description of a local computation (i.e., a program). As such, each $\pi_k(L_k)$ must specify for the protocol participant that executes it what to do in any conceivable situation. Considering its similarity to a strategy, it seems to be reasonable to relate each $\pi_k(L_k)$ to a strategy in $G_\pi(L)$.

Before going further, we need to introduce the concept of *restricted games.* Let us consider an extensive game G, and let us divide the player set P into two disjoint subsets P_{free} and P_{fix}. Furthermore, let us fix a strategy $s_j \in S_j$ for each $j \in P_{fix}$, and let us denote the vector $(s_j)_{j \in P_{fix}}$ of fixed strategies by s. The restricted game $G_{|s}$ is the extensive game that is obtained from G by restricting each $j \in P_{fix}$ to follow the fixed strategy s_j.

Note that in $G_{|s}$, each player $j \in P_{fix}$ has a single strategy to follow, namely, the strategy that is induced by the fixed strategy s_j in $G_{|s}$. This means that the outcome of $G_{|s}$ solely depends on what strategies are followed by the players in P_{free}. For any player $i \in P_{free}$ and for any strategy $s_i \in S_i$, let $s_{i|s}$ denote the strategy that s_i induces in the restricted game $G_{|s}$. Let $o_{|s}((s_{i|s})_{i \in P_{free}})$ denote the resulting outcome in $G_{|s}$ when the players in P_{free} follow the strategies in the strategy profile $(s_{i|s})_{i \in P_{free}}$.

As we said before, we want to define the concept of rational exchange in terms of a Nash equilibrium in the protocol game. Indeed, we define it in terms of a Nash equilibrium in a restricted protocol game. To be more precise, we consider the restricted protocol game that we obtain from the protocol game by restricting the trusted third party (if there is any) to follow its program faithfully (i.e., to behave correctly), and we require that the strategies that correspond to the programs of the main parties form a Nash equilibrium in this restricted protocol game. In addition, we require that no other Nash equilibrium be strongly preferable for any of the main parties in the restricted game. This ensures that the main parties are indeed interested in behaving correctly and executing their programs faithfully.

Definition 2 (Rational Exchange). *Let us consider a two-party exchange protocol $\pi(L) = \{\pi_1(L_1), \pi_2(L_2), \pi_3(L_3)\}$, where $\pi_1(L_1)$ and $\pi_2(L_2)$ are the programs for the main parties, and $\pi_3(L_3)$ is the program for the trusted third party. Furthermore, let us consider the protocol game $G_\pi(L)$ of $\pi(L)$ constructed according to the framework described in Section 3. Let us denote the strategy of player p_k that represents $\pi_k(L_k)$ within $G_\pi(L)$ by $s^*_{p_k}$ ($k \in \{1, 2, 3\}$), and let us denote the single strategy of the network by s_{net}. $\pi(L)$ is said to be* rational *iff*

1. *$(s^*_{p_1|s}, s^*_{p_2|s})$ is a Nash equilibrium in the restricted protocol game $G_{\pi|s}(L)$, where $s = (s^*_{p_3}, s_{net})$; and*

2. for any other Nash equilibrium $(s_{p_1|s}, s_{p_2|s})$ in $G_{\pi|s}(L)$, we have that

$$o_{|s}(s_{p_1|s}, s_{p_2|s}) \preceq_{p_1} o_{|s}(s^*_{p_1|s}, s^*_{p_2|s})$$
$$o_{|s}(s_{p_1|s}, s_{p_2|s}) \preceq_{p_2} o_{|s}(s^*_{p_1|s}, s^*_{p_2|s})$$

It can be proven formally that the payment protocol described in Subsection 1.1 satisfies the above definition of rational exchange, however, due to space limitation we must omit the details of the proof. The interested reader is referred to [3].

5 Relation to fair exchange

Informally, a fair exchange protocol ensures for each main party A that if A behaves correctly, then the other party B cannot get the item of A unless A gets the item of B. In terms of our game theoretic model, this property can be expressed as follows, where $s = (s^*_{p_3}, s_{net})$:

- for every strategy s_{p_1} of p_1, $y^+_{p_1}(q) = u_{p_1}(\gamma_{p_2}) \Rightarrow y^+_{p_2}(q) = u_{p_2}(\gamma_{p_1})$, where $q = o_{|s}(s_{p_1|s}, s^*_{p_2|s})$; and
- for every strategy s_{p_2} of p_2, $y^+_{p_2}(q) = u_{p_2}(\gamma_{p_1}) \Rightarrow y^+_{p_1}(q) = u_{p_1}(\gamma_{p_2})$, where $q = o_{|s}(s^*_{p_1|s}, s_{p_2|s})$.

Theorem 1. *If the protocol is closed for incomes, then fair exchange implies rational exchange.*

Proof. First, we have to prove that $(s^*_{p_1|s}, s^*_{p_2|s})$ is a Nash equilibrium in $G_{\pi|s}(L)$. Let us suppose that it is not. This means that either $s^*_{p_1|s}$ is not the best response to $s^*_{p_2|s}$, or $s^*_{p_2|s}$ is not the best response to $s^*_{p_1|s}$. Without loss of generality, let us assume that the first is the case. This means that p_1 has a strategy $s'_{p_1|s}$ such that $y_{p_1}(q^*) < y_{p_1}(q')$, where $q^* = o_{|s}(s^*_{p_1|s}, s^*_{p_2|s})$, and $q' = o_{|s}(s'_{p_1|s}, s^*_{p_2|s})$. Since q^* is the outcome when both parties behave correctly, $y^+_{p_1}(q^*) = u_{p_1}(\gamma_{p_2}) > 0$ and $y^-_{p_1}(q^*) = u_{p_1}(\gamma_{p_1}) > 0$. Therefore, $y_{p_1}(q^*) < y_{p_1}(q')$ is possible only if $y^+_{p_1}(q') = u_{p_1}(\gamma_{p_2}) > 0$ and $y^-_{p_1}(q') = 0$. However, this is impossible, because, from the properties of fair exchange, $y^+_{p_1}(q') > 0$ implies $y^+_{p_2}(q') > 0$, and from the income closed property, $y^+_{p_2}(q') > 0$ implies $y^-_{p_1}(q') > 0$.

Next, we have to prove that no other Nash equilibrium is strongly preferable for any of the players. Let us suppose the contrary, and assume that there exists a Nash equilibrium $(s'_{p_1|s}, s'_{p_2|s})$ in $G_{\pi|s}(L)$ such that $y_{p_1}(q^*) < y_{p_1}(q')$, where $q^* = o_{|s}(s^*_{p_1|s}, s^*_{p_2|s})$, and $q' = o_{|s}(s'_{p_1|s}, s'_{p_2|s})$. This is possible only if $y^+_{p_1}(q') = u_{p_1}(\gamma_{p_2}) > 0$ and $y^-_{p_1}(q') = 0$. From the income closed property, we get that $y^+_{p_1}(q') > 0$ implies $y^-_{p_2}(q') > 0$, and $y^-_{p_1}(q') = 0$ implies $y^+_{p_2}(q') = 0$. Therefore, if p_1 follows $s'_{p_1|s}$ and p_2 follows $s'_{p_2|s}$, then p_2's payoff is negative. However, p_2 can always do better, and achieve a non-negative payoff, for instance, by quitting right at the beginning of the game. This means that $s'_{p_2|s}$ is not the best response to $s'_{p_1|s}$, and thus, $(s'_{p_1|s}, s'_{p_2|s})$ cannot be a Nash equilibrium. $\square$

It is easy to see that rational exchange does not imply fair exchange. The payment protocol of Subsection 1.1, for instance, does not guarantee fairness, nevertheless, it can be proven to be a rational exchange protocol [3].

6 Related work

In [1], Asokan *et al.* model their exchange protocol as a game that is played by a misbehaving party against a correctly behaving party and a trustee according to some well defined rules. They define fairness to mean that the probability that the misbehaving party wins the game is negligible. In [5], Kremer and Raskin model non-repudiation protocols (which are strongly related to fair exchange protocols) as games in a similar way as we do. However, they use neither payoffs nor the concept of equilibrium to specify properties of the protocol. Instead, they introduce a game based alternating temporal logic for this purpose. In [9], Sandholm introduces the notion of *unenforced exchange*, and relates it to the concept of Nash equilibrium. None of these papers attempt to formalize the concept of rational exchange in general and to investigate the relationship between rational exchange and fair exchange.

7 Conclusion

In this paper, we introduced game theory as a formal framework in which exchange protocols can be modeled and their properties can be studied. We used this framework to give a formal definition for rational exchange relating it to the concept of Nash equilibrium in games. In addition, we studied the relationship between rational exchange and fair exchange. We proved that fair exchange implies rational exchange if the protocol is closed for incomes (which is usually the case), but the reverse is not true.

Our theoretical results, thus, justify the intuition that a fair exchange protocol provides stronger guarantees than a rational exchange protocol. On the other hand, fair exchange protocols are usually more complex than rational exchange protocols. Therefore, rational exchange protocols may provide an interesting solution to the exchange problem by representing a trade-off between complexity and what they achieve. In certain applications, notably in mobile e-commerce, rational exchange protocols may provide more practical solutions than fair exchange protocols.

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4), April 2000.
2. L. Buttyán. Removing the financial incentive to cheat in micropayment schemes. *IEE Electronics Letters*, 36(2):132–133, January 2000.
3. L. Buttyán. Building Blocks for Secure e-Commerce Services: Authenticated Key Transport and Rational Exchange Protocols. Ph.D. Thesis, in preparation.

4. M. Jakobsson. Ripping coins for a fair exchange. In *Proceedings of EURO-CRYPT'95*, pages 220–230, 1995.
5. S. Kremer and J.-F. Raskin. Formal verification of non-repudiation protocols – a game approach. In *Formal Methods for Computer Security*, Chicago, USA, July 2000.
6. N. Lynch. *Distributed Algorithms.* Morgan Kaufmann, 1996.
7. M. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 1994.
8. H. Pagnia, H. Vogt, and F. Gärtner. Fair exchange. manuscript, August 2001.
9. T. Sandholm. Unenforced e-commerce transactions. *IEEE Internet Computing*, pages 47–54, November-December 1997.
10. P. Syverson. Weakly secret bit commitment: applications to lotteries and fair exchange. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop*, pages 2–13, June 1998.