A Formal Model of Rational Exchange and Its Application to the Analysis of Syverson's Protocol[*]

Levente Buttyán[1,2] Jean-Pierre Hubaux[2] Srdjan Čapkun[2]

[1] Laboratory of Cryptography and Systems Security (CrySyS)
Department of Telecommunications
Budapest University of Technology and Economics
Magyar tudósok krt 2, H-1117 Budapest, Hungary

[2] Laboratory of Computer Communications and Applications
School of Computer and Communication Sciences
Swiss Federal Institute of Technology – Lausanne (EPFL)
EPFL-IC-LCA, CH-1015 Lausanne, Switzerland

buttyan@hit.bme.hu, jean-pierre.hubaux@epfl.ch, srdan.capkun@epfl.ch

April 15, 2003

Abstract

We propose a formal model of rational exchange and exchange protocols in general, which is based on game theory. In this model, an exchange protocol is represented as a set of strategies in a game that is played by the protocol parties and the network that they use to communicate with each other. Within this model, we give a formal definition for rational exchange and various other properties of exchange protocols, including fairness. In particular, rational exchange is defined in terms of a Nash equilibrium in the protocol game. We also study the relationship between rational and fair exchange, and prove that fairness implies rationality, but not vice versa. Finally, we illustrate the usage of our formal model for the analysis of existing rational exchange protocols by analyzing a protocol proposed by Syverson. We show that the protocol is rational only under the assumption that the network is reliable.

1 Introduction

Recently, new computing and networking paradigms have emerged, which are based on the concept of self-organization. The most prominent examples are peer-to-peer computing and wireless ad hoc networks. Due to their very nature, the operation of these systems is based on mechanisms that are fundamentally different from those used in traditional computing and networking systems. Of course, this applies not only to the basic mechanisms but to the security mechanisms as well [26, 31, 2, 16, 13, 7, 28, 29].

[*] © 2003 IOS Press. Accepted for publication in the Journal on Computer Security, special issue on selected papers from the 15th IEEE Computer Security Foundations Workshop.

In this context, *rational exchange* becomes particularly interesting. The concept of rational exchange has been introduced by Syverson in [27], where he describes an exchange protocol that is similar to fair exchange, but in fact, it does not provide true fairness. Syverson calls the protocol rational exchange, because it ensures that rational, self-interested parties have no reason to misbehave and deviate from it.

Although Syverson's protocol does not achieve fairness, it has a very appealing feature: it does not use a trusted third party. We started to study rational exchange exactly for this reason. In the context of the Terminodes Project[1] [14], we are concerned with the design of self-organizing wireless ad hoc networks. These are networks of mobile nodes where communication is based on multi-hop relaying. In extreme cases, such networks do not rely on a fixed infrastructure at all (e.g., military and rescue operations). In less extreme cases, ad hoc networks are considered as extensions to an already established fixed infrastructure (e.g., multi-hop cellular networks [18, 30]). In both cases, the use of exchange protocols that rely on a trusted third party to achieve fairness is problematic. The reason is that, in the infrastructureless case, the existence of a trusted third party simply cannot be assumed, while in the other case, even if a trusted third party is present, there is no guarantee that the nodes can access it in a timely manner due to frequent and unpredictable disconnections from the fixed infrastructure. Although fair exchange protocols that do not rely on a trusted third party do exist (e.g., gradual secret release schemes [11] and probabilistic protocols [20]), they are highly inefficient in the sense that they require a high number of messages to be exchanged in order to achieve an acceptable level of fairness. As a consequence, they are not suitable for applications in wireless ad hoc networks, where the number of transmissions should be minimized due to the limited available bandwidth and in order to reduce the energy consumption (i.e., save battery power and reduce interference) of the nodes. Rational exchange, on the other hand, seems to be a promising alternative to solve the problem, because rational exchange protocols may not use a trusted third party, they require only a few messages to exchange, and they still provide some guarantees with respect to fairness (their relation to the property of fairness will be clarified later in this paper).

In this paper, we give a formal definition for rational exchange. The value of a formal definition is threefold:

- First, attempting to give a formal definition itself helps to better understand the concept, which is a prerequisite for any design.
- Second, it requires the construction of a mathematical model, in which other, similar concepts, such as fair exchange, can also be defined and compared to rational exchange. Such a comparison may also help the better understanding of rational exchange. Here we note that protocols with a flavor similar to that of the Syverson protocol have already been proposed earlier (e.g., [15]), but they were inappropriately called fair exchange. A precise study on the relationship of the two concepts helps to clarify this confusion.
- Third, a formal definition is indispensable to the rigorous verification of rational exchange protocols.

The mathematical model, in which we will develop our formal definition is based on game theory [21]. Game theory is a set of analytical tools developed to study situations in which self-interested parties (which want to maximize their own benefits) interact with each other according to certain rules. Since exactly this kind of situations occur in exchange protocols, game theory appears to be a natural choice.

[1]http://www.terminodes.org/

Thus, we model the situation in which parties of a given exchange protocol find themselves as a game. We call this game the protocol game. The protocol game encodes all the possible interactions of the protocol parties. The protocol parties are modeled as players. The protocol itself (as a set of rules) is represented as a set of strategies (one strategy for each protocol party). Misbehavior means that a protocol party follows a strategy that is different from its prescribed strategy.

We define the concept of rational exchange in terms of properties of the protocol game and the prescribed strategies of the protocol parties. More precisely, we have been inspired by the striking similarity between rational exchange as defined informally by Syverson and the concept of Nash equilibrium in games. Therefore, we define rational exchange formally in terms of a Nash equilibrium in the protocol game.

Our model is sufficiently rich to permit the definition of other properties of exchange protocols as well. More specifically, we can also define fairness. Representing the concepts of rational exchange and fair exchange in the same model allows us to study their relationships. In particular, we prove that fairness implies rationality (assuming that the protocol satisfies certain additional requirements), but the reverse is not true in general. Thus, the result that we obtain from the model justifies the intuition that fairness is a stronger requirement than rationality.

Finally, defining a formal model for exchange protocols and giving a formal definition for rational exchange in this model allows us to rigorously verify existing rational exchange protocols. In order to illustrate this, we formally prove that the Syverson protocol satisfies our definition of rational exchange under the assumption that the communication between the protocol parties is reliable. However, if we relax this assumption, then rationality is lost.

To the best of our knowledge, we are the first who formalized the concept of rational exchange in its full generality, studied its relation to fair exchange, and provided rigorous proofs of rationality for existing rational exchange protocols. Although game theory has already been applied in the context of exchange protocols (see e.g., [23, 17]), we are not aware of any formal model with the same precision and generality as our protocol game model. Preliminary results of our work appeared earlier in [6, 9].

The outline of the paper is the following: In Section 2, we briefly introduce some basic notions from game theory that we will use in the development of our model. We present a general framework for the modeling of exchange protocols as games in Section 3. Based on this, in Section 4, we formally define rational exchange and various other properties of exchange protocols including fairness. We study the relationship between rational exchange and fair exchange in the same section. In Section 5, we illustrate the usage of our model for the analysis of existing rational exchange protocols by analyzing the Syverson protocol. Finally, we report on some related work in Section 6, and conclude the paper in Section 7.

2 Preliminaries

In this section, we briefly introduce some notions from game theory that we will use in the paper.

2.1 Extensive games

An *extensive game* is a tuple

$$\langle P, A, Q, p, (\mathcal{I}_i)_{i \in P},\ (\preceq_i)_{i \in P} \rangle$$

where

- P is a set of *players*;

- A is a set of *actions*;
- 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 $\{a \in A : q.a \in Q\}$ of *available actions* after q.
- p is a *player function* that assigns a player in P to every non-terminal action sequence $q \in Q \setminus Z$ (the interpretation is that player $p(q)$ has to move after action sequence q);
- $\mathcal{I}_i$ 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* $I_i \in \mathcal{I}_i$;
- $\preceq_i$ 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 $q \preceq_i q'$, 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 $i \in P$, $q \preceq_i q'$ iff $y_i(q) \leq y_i(q')$.

A finite extensive game can conveniently be represented as a tree, where 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 and may be labeled with payoff vectors to represent the preference relations of the players.

Conceptually, an infinite game (i.e., a game that has infinite action sequences) can also be thought of as a tree. In this case, the infinite action sequences of the game are represented by infinite paths starting from the root.

2.2 Strategy

A *strategy of player* i is defined as a function s_i that assigns an action in $A(q)$ to each non-terminal action sequence q that is in the domain of s_i, 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. The domain $\mathrm{dom}(s_i)$ of s_i contains only those non-terminal action sequences q for which $p(q) = i$ *and* q *is consistent with the moves prescribed by* s_i. Formally, we can define $\mathrm{dom}(s_i)$ in an inductive way as follows: A non-terminal action sequence $q = (a_k)_{k=1}^{w}$ is in $\mathrm{dom}(s_i)$ iff $p(q) = i$ and

- either there is no $0 \leq v < w$ such that $p((a_k)_{k=1}^{v}) = i$;
- or for all $0 \leq v < w$ such that $p((a_k)_{k=1}^{v}) = i$, $(a_k)_{k=1}^{v}$ is in $\mathrm{dom}(s_i)$ and $s_i((a_k)_{k=1}^{v}) = a_{v+1}$.

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

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

This means that if every player i other than j follows s_i^*, then player j is not motivated to deviate from s_j^*, because she does not gain anything by doing so. It is possible that a game has multiple Nash equilibria.

3 Protocol games

Game theory in general, and the above introduced notions in particular, will serve as the basis of our model of rational exchange. We describe this model in two steps: First, in this section, we introduce a general framework for the construction of games from exchange protocols. We refer to these games as *protocol games*. The protocol game of an exchange protocol is intended to model all the possible interactions of the (potentially misbehaving) protocol parties. The correct behavior of each party is represented by a particular strategy within the protocol game. Second, in the next section, we define rational exchange formally as a particular property that the strategies representing the correct behavior of the protocol parties should satisfy.

We should note that we consider only two-party exchange protocols (i.e., protocols that involve only two main parties and possibly a trusted third party) for two reasons. First, we want to make the presentation easier. Second, most of the exchange protocols proposed in the literature are two-party exchange protocols. However, our model could be extended to multi-party exchange protocols as well.

3.1 System model

We assume that the network that is used by the protocol participants to communicate with each other is reliable, which means that it delivers messages to their intended destinations within a constant time interval. Such a network allows the protocol participants to interact in a synchronous fashion. We will model this by assuming that the protocol participants interact with each other in *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 [19], where the same model is used to study the properties of distributed algorithms in a synchronous network system.

As we mentioned in the Section 1, our work was motivated by the use of rational exchange in wireless ad hoc networks. Clearly, the synchronous model defined above is far from being realistic for such networks in general. Nevertheless, it makes sense to start the investigation with a simpler model as this may pave the way to the more general asynchronous case. One step in this direction is presented in [4], where we sketch how the synchrony assumption could be relaxed and how asynchronous systems could be modeled as games.

In addition, there are applications where the synchronous model defined above is not so unrealistic. Consider for instance two neighboring nodes of an ad hoc network that want to perform some transaction with each other (e.g., execute an exchange protocol). Transactions between neighbors may be common in certain types of ad hoc networks (see for instance [5, 8]). In this case, there are better reasons to assume bounds on the message delivery delays, because the nodes communicate directly and not via intermediate forwarding nodes. Moreover, the underlying medium access control scheme may also provide mechanisms (e.g., the optional RTS/CTS handshake in IEEE 802.11) that makes the communication between neighboring nodes more reliable.

3.2 Limitations on misbehavior

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 have nothing to do with the protocol being studied. 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 impose on protocol participants is that they can send only messages that are *compatible* with the protocol. We make this more precise in the following paragraph.

We consider an exchange protocol to be a description π of a distributed computation that consists of a set $\{\pi_1, \pi_2, \ \ldots\}$ of descriptions of local computations. For brevity, we call these descriptions of local computations *programs*. Each program π_k is meant to be executed by a protocol participant. Typically, each π_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 conditions is provided (or a timeout occurs). We call a message m compatible with π_k if the local computation described by π_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 π_k by M_{π_k}. Then, the set of messages that are compatible with the protocol is defined as $M_\pi = \cup_k M_{\pi_k}$.

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. This also means that the protocol participants may alter the prescribed order of the protocol messages (if this is not prevented deliberately by the design of the protocol).

On the other hand, we note that our model does not allow the protocol parties to run multiple instances of the protocol in parallel (i.e., we do not consider interleaving attacks), and to eavesdrop or modify messages sent between other parties of the protocol.

3.3 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. However, usually, the payoffs for the main parties depend on the state of the trusted third party, and it is easier to handle the state transitions of the trusted third party if we explicitly model it as a player. In addition, modeling the trusted third party in the same way as we model the other protocol participants leads to a more uniform model. After all, the trusted third party *is* a protocol participant. We will make the distinction between the trusted third party and the potentially misbehaving main parties of the protocol in another way: we restrict the player that represents the trusted third party to follow a particular strategy (the one that represents the correct behavior), whereas we allow the players that represent the potentially misbehaving main parties to choose among several strategies.

As we mentioned before, we assume that the protocol participants interact in synchronous rounds, where every message sent in the first phase of a round is delivered in the second phase of the same round. It might again seem that it is useless to model the network explicitly as a player, because the only action it can perform is the delivery of the messages that were sent in the current round, and therefore, it does not have choices. Nevertheless, we represent the network explicitly as a player. The reason is that it seems to be easier to present the model if we explicitly include the message delivery actions, because they clearly identify the second phases of the rounds, and thus, the points where the states of the players change as the result of obtaining (partial) information about the actions performed by the other players. In addition, modeling the network explicitly as a player makes it easier to extend our model with unreliable networks, because such networks can be modeled as real players that can choose between delivering a message or further delaying it.

3.4 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 $\textsf{\textit{snd}}(m, j)$ is generated for player $i \in P'$ when she submits a message $m \in M_\pi$ with intended destination $j \in P'$ to the network, and the receive event $\textsf{\textit{rcv}}(m)$ is generated for player $i \in P'$ when the network delivers a message $m \in M_\pi$ 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 $\langle \alpha_i(q), H_i(q), r_i(q) \rangle$, where

- $\alpha_i(q) \in \{\textsf{true}, \textsf{false}\}$ is a boolean, which is $\textsf{true}$ iff player i is still active after action sequence q (i.e., she did not quit the protocol);
- $H_i(q) \subseteq E \times \mathbb{N}$ 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;
- $r_i(q) \in \mathbb{N}$ is a non-negative integer that represents the round number for player i after action sequence q.

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

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

3.5 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 $m \in M_\pi$ 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 $m' \in M_\pi$ 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 $j \neq i$.

While the formal derivation of the condition tags attached to the messages are currently not supported by our method, we had no particular problems deriving them for the protocols that we have analyzed. The reason may be that each of the logical formulae is concerned with a *single* message, or more precisely the conditions upon which that message can be sent by a given protocol participant. Nevertheless, in our future work, we may develop a more systematic approach for this purpose in order to avoid possible errors that this informal step might introduce in the analysis.

Now, let us consider an action sequence q, after which player $i \in P'$ has to move. There are two special actions, called $\textsf{idle}_i$ and $\textsf{quit}_i$, which are always available for i after q. In addition to these special actions, player i can choose a send action of the form $\textsf{send}_i(M)$, where M is a subset of the set $M_i(\Sigma_i(q))$ of messages that i is able to send in her current local state. Formally, we define $M_i(\Sigma_i(q))$ as

$$M_i(\Sigma_i(q)) = \{(m, j) : m \in M_\pi,\ \phi_i^m(\Sigma_i(q)) = \textsf{true},\ j \in P' \setminus \{i\}\}$$

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

$$A_i(\Sigma_i(q)) = \{\textsf{idle}_i,\ \textsf{quit}_i\} \cup \{\textsf{send}_i(M) : M \subseteq M_i(\Sigma_i(q))\}$$

Note that $\mathsf{send}_i(\emptyset) \in A_i(\Sigma_i(q))$. By convention, $\mathsf{send}_i(\emptyset) = \mathsf{idle}_i$.

Let us consider now an action sequence q, after which the network has to move. Since the network is assumed to be reliable, it should deliver every message that was submitted to it in the current round. This means that there is only one action, called $\mathsf{deliver}_{net}$, that is available for the network after q, which means the delivery of all messages in the network buffer. Thus,

$$A_{net}(\Sigma_{net}(q)) = \{\mathsf{deliver}_{net}\}$$

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

- If a player $i \in P'$ performs the action idle_i, then the state of every player $j \in P$ remains the same as before.

  Formally: for any action sequence q, after which player $i \in P'$ has to move, we have that

  $$\Sigma_j(q.\mathsf{idle}_i) = \Sigma_j(q)$$

  for every $j \in P$.

- 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.

  Formally: for any action sequence q, after which player $i \in P'$ has to move, we have that

  $$\begin{aligned} \alpha_i(q.\mathsf{quit}_i) &= \mathsf{false} \\ H_i(q.\mathsf{quit}_i) &= H_i(q) \\ r_i(q.\mathsf{quit}_i) &= r_i(q) \end{aligned}$$

  and for every $j \in P \setminus \{i\}$,

  $$\Sigma_j(q.\mathsf{quit}_i) = \Sigma_j(q)$$

- If a player $i \in P'$ performs an action $\mathsf{send}_i(M)$ such that $M \neq \emptyset$, then the messages in M 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.

  Formally: for any action sequence q, after which player $i \in P'$ has to move, and for any available send action $\mathsf{send}_i(M) \in A_i(\Sigma_i(q))$ such that $M \neq \emptyset$, we have that

  $$\begin{aligned} \alpha_i(q.\mathsf{send}_i(M)) &= \alpha_i(q) \\ H_i(q.\mathsf{send}_i(M)) &= H_i(q) \cup \{(\mathit{snd}(m,j), r_i(q)) : (m,j) \in M\} \\ r_i(q.\mathsf{send}_i(M)) &= r_i(q) \end{aligned}$$

  $$M_{net}(q.\mathsf{send}_i(M)) = M_{net}(q) \cup \{(m,i,j) : (m,j) \in M\}$$

  and for every $j \in P \setminus \{i, net\}$,

  $$\Sigma_j(q.\mathsf{send}_i(M)) = \Sigma_j(q)$$

- If the network performs the action $\mathsf{deliver}_{net}$, then for every message in the network buffer, the appropriate receive event is generated for the intended destination of the message if it is still active. Then, every message is removed from the network buffer, and the round number of every active player is increased by one.

  Formally: for any action sequence q, after which the network has to move, we have that

$$M_{net}(q.\mathsf{deliver}_{net}) \quad = \quad \emptyset$$

  and for every $i \in P'$,

  - if $\alpha_i(q) = \mathsf{true}$, then

$$\begin{aligned} \alpha_i(q.\mathsf{deliver}_{net}) &= \alpha_i(q) \\ H_i(q.\mathsf{deliver}_{net}) &= H_i(q) \cup \{(\mathsf{rcv}(m), r_i(q)) : \exists j \in P' : (m, j, i) \in M_{net}(q)\} \\ r_i(q.\mathsf{deliver}_{net}) &= r_i(q) + 1 \end{aligned}$$

  - otherwise

$$\Sigma_i(q.\mathsf{deliver}_{net}) \quad = \quad \Sigma_i(q)$$

3.6 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 formal, let us denote the set of players that are still active after action sequence q and have an index larger than v by $P'(q, v)$ (i.e., $P'(q, v) = \{p_k : p_k \in P',\ \alpha_{p_k}(q) = \mathsf{true},\ k > v\}$). Furthermore, let us denote the smallest index in $P'(q, v)$ by $k_{min}(q, v)$.

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
  1. $q.a \in Q$ for every $a \in A_{p_v}(\Sigma_{p_v}(q))$;
  2. 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
  1. $q.a \in Q$ for the single action $a = \mathsf{deliver}_{net} \in A_{net}(\Sigma_{net}(q))$;
  2. 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, and thus, p is not defined in $q.a$.

3.7 Payoffs

Now, we describe how the payoffs are determined. Let us consider the two main parties p_1 and p_2 of the protocol, and the items γ_{p_1} and γ_{p_2} that they want to exchange. We denote the values that γ_{p_1} is worth to p_1 and p_2 by $u^-_{p_1}$ and $u^+_{p_2}$, respectively. Similarly, the values that γ_{p_2} is worth to p_1 and p_2 are denoted by $u^+_{p_1}$ and $u^-_{p_2}$, respectively (see also Table 1).

	γ_{p_1}	γ_{p_2}
p_1	$u_{p_1}^-$	$u_{p_1}^+$
p_2	$u_{p_2}^+$	$u_{p_2}^-$

Table 1: The values that the items to be exchanged are worth to the protocol parties

Intuitively, u_i^+ and u_i^- can be thought of as a potential gain and a potential loss of player $i \in \{p_1, p_2\}$ in the game. In practice, it may be difficult to quantify u_i^+ and u_i^-. However, our approach does not depend on the exact values; we require only that $u_i^+ > u_i^-$ for both $i \in \{p_1, p_2\}$, which we consider to be a necessary condition for the exchange to take place at all. In addition, we will assume that $u_i^- > 0$.

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 *gain* and $y_i^-(q)$ the *loss* of player i, and define them as follows:

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

and

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

where $\phi_i^+(q)$ 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 ($j \neq i$), and $\phi_i^-(q) = \mathsf{true}$ iff i loses control over γ_i in q. A typical example would be $\phi_i^+(q) = (\exists r : (\mathit{rcv}(m), r) \in H_i(q))$, where we assume that m is the only message in M_π that contains γ_j.

Note that according to our model, the payoff $y_i(q)$ of player i can take only four possible values: u_i^+, $u_i^+ - u_i^-$, 0, and $-u_i^-$ 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.

3.8 Protocol vs. protocol game

Although the protocol game is constructed from the description of the protocol, it represents more than the protocol itself, because it also encodes the possible misbehavior of the parties, which is not specified in the protocol (at least not explicitly). Recall that a protocol is considered here to be a set of programs $\pi = \{\pi_1, \pi_2, \ldots\}$. Each program π_i must specify for the protocol participant that executes it what to do in any conceivable situation. In this sense, a program is very similar to a strategy. Therefore, we model the protocol itself as a set of strategies (one strategy for each program) in the protocol game. We will denote the strategy that corresponds to π_i by s_i^*.

4 Formal definition of rational exchange and other properties

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.

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 $j \in P_{\mathit{fix}}$, and let us denote the vector $(s_j)_{j \in P_{\mathit{fix}}}$ of fixed strategies by $\bar{s}_{\mathit{fix}}$. The restricted game $G_{|\bar{s}_{\mathit{fix}}}$ is the extensive game that is obtained from G by restricting each $j \in P_{\mathit{fix}}$ to follow the fixed strategy s_j.

Note that in $G_{|\bar{s}_{\mathit{fix}}}$, only the players in P_{free} can have several strategies; the players in P_{fix} are bound to the fixed strategies in $\bar{s}_{\mathit{fix}}$. This means that the outcome of $G_{|\bar{s}_{\mathit{fix}}}$ solely depends on what strategies are followed by the players in P_{free}. In other words, the players in P_{fix} become *pseudo* players, which are present, but do not have any influence on the outcome of the game.

For any player $i \in P_{\mathit{free}}$ and for any strategy $s_i \in S_i$ of player i, let $s_{i|\bar{s}_{\mathit{fix}}}$ denote the strategy that s_i induces in the restricted game $G_{|\bar{s}_{\mathit{fix}}}$. In addition, let us denote the resulting outcome in $G_{|\bar{s}_{\mathit{fix}}}$ when the players in P_{free} follow the strategies in the strategy profile $(s_{i|\bar{s}_{\mathit{fix}}})_{i \in P_{\mathit{free}}}$ by $o_{|\bar{s}_{\mathit{fix}}}((s_{i|\bar{s}_{\mathit{fix}}})_{i \in P_{\mathit{free}}})$.

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 have indeed no rational interest in deviating from the faithful execution of their programs.

Definition 1 *Let us consider a two-party exchange protocol $\pi = \{\pi_1, \pi_2, \pi_3\}$, where π_1 and π_2 are the programs for the main parties, and π_3 is the program for the trusted third party (if there is any). Furthermore, let us consider the protocol game G_π of π constructed according to the framework described in Section 3. Let us denote the strategy of player p_k that belongs to the faithful execution of π_k within G_π by $s^*_{p_k}$ ($k \in \{1, 2, 3\}$), the single strategy of the network by s^*_{net}, and the strategy vector $(s^*_{p_3}, s^*_{net})$ by $\bar{s}$.*

- ***Rationality:*** *π is said to be* rational *iff*
  - *$(s^*_{p_1|\bar{s}}, s^*_{p_2|\bar{s}})$ is a Nash equilibrium in the restricted protocol game $G_{\pi|\bar{s}}$; and*
  - *both p_1 and p_2 prefer the outcome of $(s^*_{p_1|\bar{s}}, s^*_{p_2|\bar{s}})$ to the outcome of any other Nash equilibrium in $G_{\pi|\bar{s}}$.*

Besides rationality, our model allows us to define other properties of exchange protocols as well. Most importantly, we can give a formal definition for the properties of *fairness, effectiveness,* and *termination*. Informally, fairness means that if a party A behaves correctly, then the other party B cannot get the item of A unless A gets the item of B. Effectiveness requires that if both parties behave correctly, then both have access to the other's item when the protocol is completed. Finally, termination means that each correctly behaving party will eventually terminate execution.

Definition 2 *Let us consider the notation introduced in Definition 1.*

- ***Fairness:*** *π is said to be* fair *iff*
  - *for every strategy $s_{p_1|\bar{s}}$ of p_1, $y^+_{p_1}(q) > 0$ implies $y^+_{p_2}(q) > 0$, where $q = o_{|\bar{s}}(s_{p_1|\bar{s}}, s^*_{p_2|\bar{s}})$; and*
  - *for every strategy $s_{p_2|\bar{s}}$ of p_2, $y^+_{p_2}(q) > 0$ implies $y^+_{p_1}(q) > 0$, where $q = o_{|\bar{s}}(s^*_{p_1|\bar{s}}, s_{p_2|\bar{s}})$.*
- ***Effectiveness:*** *π is said to be* effective *iff $y^+_{p_1}(q^*) > 0$ and $y^+_{p_2}(q^*) > 0$, where $q^* = o_{|\bar{s}}(s^*_{p_1|\bar{s}}, s^*_{p_2|\bar{s}})$.*
- ***Termination:*** *π is said to be* terminating *iff*
  - *for every strategy $s_{p_1|\bar{s}}$ of p_1, there exists a finite prefix q' of q, such that $\alpha_{p_2}(q') = \mathsf{false}$, where $q = o_{|\bar{s}}(s_{p_1|\bar{s}}, s^*_{p_2|\bar{s}})$; and*
  - *for every strategy $s_{p_2|\bar{s}}$ of p_2, there exists a finite prefix q' of q, such that $\alpha_{p_1}(q') = \mathsf{false}$, where $q = o_{|\bar{s}}(s^*_{p_1|\bar{s}}, s_{p_2|\bar{s}})$.*

All the properties above are defined in the restricted game, where the trusted third party is restricted to follow its program faithfully (i.e., to behave correctly). Fairness requires that if a player follows the strategy that corresponds to the faithful execution of her program, then the other player can have a positive gain only if the well behaving player also has a positive gain. Recall that having a positive gain represents a state where the player has access to the expected item. So our formal definition corresponds to the informal characterization of fairness. Effectiveness requires that if both players follow the strategy that corresponds to the faithful execution of their programs, then the outcome will be an action sequence in which the gain of both players is positive (this represents a state, where both players have access to the expected items). Finally, termination requires that if a player follows the strategy that corresponds to the faithful execution of her program (i.e., she behaves correctly), then no matter what strategy is played by the other player, the well behaving player will terminate computation and reach an inactive state (i.e., she will perform the quit action) in a finite number of rounds.

In addition to the above definitions, we also define two other properties called *gain closed property* and *safe back out property* that we will use later. The gain closed property requires that if a party A gains access to the item of the other party B, then B loses control over the same item. The safe back out property requires that if a party abandons the exchange right at the beginning without doing anything else, then she will not lose control over her item (i.e., it is safe to back out of the exchange). All the protocols that we are aware of satisfy these properties; we need to define them for technical reasons only.

Definition 3 *Let us consider the notation introduced in Definition 1.*

- ***Gain closed property:*** *π is said to be* gain closed *iff for every terminal action sequence q of $G_{\pi|\bar{s}}$ we have that $y^+_{p_1}(q) > 0$ implies $y^-_{p_2}(q) > 0$ and $y^+_{p_2}(q) > 0$ implies $y^-_{p_1}(q) > 0$.*
- ***Safe back out property:*** *Let $Q' = \{(a_k)^w_{k=1} \in Q_{|\bar{s}} : p_{|\bar{s}}((a_k)^w_{k=1}) = p_1,\ \nexists v < w : p_{|\bar{s}}((a_k)^v_{k=1}) = p_1\}$, and let $s^0_{p_1|\bar{s}}$ be the strategy of p_1 that assigns quit_{p_1} to every action sequence in Q'. Similarly, let $Q'' = \{(a_k)^w_{k=1} \in Q_{|\bar{s}} : p_{|\bar{s}}((a_k)^w_{k=1}) = p_2,\ \nexists v < w : p_{|\bar{s}}((a_k)^v_{k=1}) = p_2\}$, and let $s^0_{p_2|\bar{s}}$ be the strategy of p_2 that assigns quit_{p_2} to every action sequence in Q''. π satisfies the* safe back out *property iff*
  - *for every strategy $s_{p_1|\bar{s}}$ of p_1, $y^-_{p_2}(q) = 0$, where $q = o_{|\bar{s}}(s_{p_1|\bar{s}}, s^0_{p_2|\bar{s}})$; and*
  - *for every strategy $s_{p_2|\bar{s}}$ of p_2, $y^-_{p_1}(q) = 0$, where $q = o_{|\bar{s}}(s^0_{p_1|\bar{s}}, s_{p_2|\bar{s}})$.*

4.1 Relationship between rational and fair exchange

Proposition 1 *If the protocol satisfies the effectiveness, gain closed, and safe back out properties, then fairness implies rationality.*

Proof: First, we have to prove that $(s^*_{p_1|\bar{s}}, s^*_{p_2|\bar{s}})$ is a Nash equilibrium in $G_{\pi|\bar{s}}$ where $\bar{s} = (s^*_{p_3}, s^*_{net})$. Let us suppose that it is not. This means that either $s^*_{p_1|\bar{s}}$ is not the best response to $s^*_{p_2|\bar{s}}$, or $s^*_{p_2|\bar{s}}$ is not the best response to $s^*_{p_1|\bar{s}}$. Without loss of generality, we can assume that the first is the case. This means that p_1 has a strategy $s'_{p_1|\bar{s}}$ such that playing $s'_{p_1|\bar{s}}$ against $s^*_{p_2|\bar{s}}$ yields a higher payoff for p_1 than the payoff that she gets if she plays $s^*_{p_1|\bar{s}}$. In other words, $y_{p_1}(q^*) < y_{p_1}(q')$, where $q^* = o_{|\bar{s}}(s^*_{p_1|\bar{s}}, s^*_{p_2|\bar{s}})$, and $q' = o_{|\bar{s}}(s'_{p_1|\bar{s}}, s^*_{p_2|\bar{s}})$. Since q^* is the outcome when both parties behave correctly and, by assumption, the protocol is effective, we have that $y^+_{p_1}(q^*) > 0$ and $y^+_{p_2}(q^*) > 0$. In addition, since the protocol is also gain closed, we get that $y^-_{p_1}(q^*) > 0$ and $y^-_{p_2}(q^*) > 0$. This means that $y_{p_1}(q^*) < y_{p_1}(q')$ is possible only if $y^+_{p_1}(q') > 0$ and $y^-_{p_1}(q') = 0$ hold. However, this is impossible, because, from the fairness property, $y^+_{p_1}(q') > 0$ implies $y^+_{p_2}(q') > 0$, and from the gain 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 $(s'_{p_1|\bar{s}}, s'_{p_2|\bar{s}})$ in $G_{\pi|\bar{s}}$ such that one of the players, say p_1, has a higher payoff if $(s'_{p_1|\bar{s}}, s'_{p_2|\bar{s}})$ is played than if $(s^*_{p_1|\bar{s}}, s^*_{p_2|\bar{s}})$ is played. This means that $y_{p_1}(q^*) < y_{p_1}(q')$, where $q^* = o_{|\bar{s}}(s^*_{p_1|\bar{s}}, s^*_{p_2|\bar{s}})$, and $q' = o_{|\bar{s}}(s'_{p_1|\bar{s}}, s'_{p_2|\bar{s}})$. For similar reasons as before, $y_{p_1}(q^*) < y_{p_1}(q')$ is possible only if $y^+_{p_1}(q') > 0$ and $y^-_{p_1}(q') = 0$ hold. Now, from the gain 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, the payoff $y_{p_2}(q')$ of p_2 in q' is negative. However, since the protocol has the safe back out property, p_2 can always do better, and achieve a non-negative payoff by not participating in the exchange at all (i.e., quitting at the beginning of the protocol without doing anything). This means that $s'_{p_2|\bar{s}}$ is not the best response to $s'_{p_1|\bar{s}}$, and thus, $(s'_{p_1|\bar{s}}, s'_{p_2|\bar{s}})$ cannot be a Nash equilibrium. □

We have just proved that fairness implies rationality. However, the reverse is not true in general. In the next section, we will prove that the protocol proposed by Syverson in [27] is rational, but it is clear that it does not provide fairness.

Our result shows that fairness is indeed a stronger requirement than rationality. Therefore, one expects that rational exchange protocols are less complex and/or have fewer system requirements than fair exchange protocols. This suggests that rational exchange can be viewed as a trade-off between complexity and true fairness, and as such, it may provide interesting solutions to the exchange problem in applications where fair exchange would be impossible or inefficient (e.g., in infrastructureless ad hoc networks).

5 Analysis of the Syverson protocol

In this section, we analyze the rational exchange protocol proposed by Syverson in [27] using our protocol game model and our formal definition of rationality. The Syverson protocol is illustrated in Figure 1, where A and B denote the two protocol participants; k_A^{-1} and k_B^{-1} denote their private keys; $item_A$ and $item_B$ denote the items that they want to exchange[2]; dsc_A denotes the description of $item_A$; and k denotes a randomly chosen secret key. In addition, enc is a symmetric-key encryption

[2]We took the liberty to replace *Payment* in the original protocol description with $item_B$ in our description. This change makes the protocol more general, and it has no effect on the properties of the protocol.

$$
\begin{array}{ll}
A \to B: & m_1 = (dsc_A,\ enc(k, item_A),\ w(k),\ sig(k_A^{-1}, (dsc_A, enc(k, item_A), w(k)))) \\
B \to A: & m_2 = (item_B,\ m_1,\ sig(k_B^{-1}, (item_B, m_1))) \\
A \to B: & m_3 = (k,\ m_2,\ sig(k_A^{-1}, (k, m_2)))
\end{array}
$$

Figure 1: Syverson's rational exchange protocol

function that takes as input a key κ and a message μ, and outputs the encryption of μ with κ; sig is a signature generation function that takes a private key κ_i^{-1} and a message μ, and returns a digital signature on μ generated with κ_i^{-1}; and w is a *temporarily secret commitment* function.

The idea of temporarily secret commitment is similar to that of commitment. The difference is that the secrecy of the commitment is breakable within acceptable bounds on time (computation). More precisely, if w is a temporarily secret commitment function, then given $w(x)$, one can determine the bit string x in time t, where t lies between acceptable lower and upper bounds. For details on how to implement such a function, the reader is referred to [27].

In the first step of the protocol, A generates a random secret key k; encrypts $item_A$ with k; computes the temporarily secret commitment $w(k)$; generates a digital signature on the description dsc_A of $item_A$, the encryption of $item_A$, and the commitment $w(k)$; and sends message m_1 to B.

When B receives m_1, she verifies the digital signature and the description dsc_A of the expected item. If B is satisfied, then she sends message m_2 to A. m_2 contains $item_B$, the received message m_1, and a digital signature of B on these elements.

When A receives m_2, she verifies the digital signature, checks if the received message contains m_1, and checks if the received item matches the expectations. If she is satisfied, then she sends the key k to B in message m_3, which also contains the received message m_2 and the digital signature of A on the message content.

When B receives m_3, she verifies the digital signature, and checks if the received message contains m_2. Then, B decrypts the encrypted item in m_1 (also received as part of m_3) with the key received in m_3.

5.1 Observations

When B receives m_1, she has something that either turns out to be what she wants or evidence that A cheated, which can be used against A in a dispute. At this point, B might try to break the commitment $w(k)$ in order to obtain k and then $item_A$. However, this requires time. If $item_A$ does not lose its value in time, and the inconvenience of the delay (and the computation) is not an issue for B, then breaking the commitment is indeed the best strategy for B. The Syverson protocol should not be used in this case. So it is assumed that $item_A$ has a diminishing value in time (e.g., it could be a short term investment advice), and that it is practically worth nothing by the time at which B can break the commitment [27]. Therefore, B is interested in continuing the protocol by sending m_2 to A.

When A receives m_2, she might not send m_3 at all or for a long time. If A does not lose anything until B gets access to $item_A$, then this is indeed a good strategy for A. If this is the case, then the Syverson protocol should not be used. So it is assumed that A loses control over $item_A$ by sending it to B in m_1, even if she sends it only in an encrypted form[3]. In this case, A does not gain anything by not sending m_3 to B promptly.

[3]More precisely, it is assumed that A loses the value that $item_A$ represents for her when sending $item_A$ in m_1 even

Note, however, that A may send some garbage instead of the encrypted item in m_1. A deterrent against this is that the commitment can be broken anyhow, which means that the misbehavior of A can be discovered by B. In addition, since m_1 is signed by A, it can be used against A in a dispute. If some punishment (the value of which greatly exceeds the value of the exchanged items) for the misbehavior can be enforced, then it is not in the interest of A to cheat. Note that this punishment could be enforced externally (e.g., by law enforcement).

5.2 The set of compatible messages

In order to define the set of messages that are compatible with the protocol, we must first introduce some further notation:

- the public keys of A and B are denoted by k_A and k_B, respectively;
- vfy is a signature verification function that takes a public key κ_i, a message μ, and a signature σ, and returns true if σ is a valid signature on μ that can be verified with κ_i, otherwise it returns false;
- dsc_B denotes the description of $item_B$;
- fit is a function that takes an item γ and an item description δ as inputs, and returns true if δ matches γ, otherwise it returns false; and
- dec denotes the decryption function that belongs to enc, which takes a key κ and a ciphertext ε, and returns the decryption of ε with κ.

Next, we reconstruct the programs of the protocol participants:

```
$\pi_A(A, k_A^{-1}, B, k_B, item_A, dsc_A, dsc_B, k) =$
    1. compute $\varepsilon = enc(k, item_A)$
    2. compute $\omega = w(k)$
    3. compute $\sigma = sig(k_A^{-1}, (dsc_A, \varepsilon, \omega))$
    4. send $(dsc_A, \varepsilon, \omega, \sigma)$ to $B$
    5. wait until timeout or
       a message $m = (\gamma, \mu, \sigma')$ arrives such that
           - $\mu = (dsc_A, \varepsilon, \omega, \sigma)$
           - $fit(\gamma, dsc_B) =$ true
           - $vfy(k_B,\ (\gamma, \mu),\ \sigma') =$ true
    6. if timeout then go to step 9
    7. compute $\sigma'' = sig(k_A^{-1}, (k, m))$
    8. send $(k, m, \sigma'')$ to $B$
    9. exit

$\pi_B(B, k_B^{-1}, A, k_A, item_B, dsc_A) =$
    1. wait until timeout or
       a message $m = (\delta, \varepsilon, \omega, \sigma)$ arrives such that
```

though m_1 is encrypted. An example would be when $item_A$ is a result of some computation that has a cost for A. In this case, the mere fact that A sends $item_A$ in m_1 means that A has already performed the computation, and thus, lost something, although B has not gained anything yet.

```
        - $\delta = dsc_A$
        - $vfy(k_A,\ (\delta, \varepsilon, \omega),\ \sigma) = \text{true}$
2. if timeout then go to step 6
3. compute $\sigma' = sig(k_B^{-1}, (item_B, m))$
4. send $(item_B, m, \sigma')$ to $A$
5. wait until timeout or
   a message $m' = (\kappa, \mu, \sigma'')$ arrives such that
        - $\mu = (item_B, m, \sigma')$
        - $fit(dec(\kappa, \varepsilon), dsc_A) = \text{true}$
        - $vfy(k_A,\ (\kappa, \mu),\ \sigma'') = \text{true}$
6. exit
```

Once the programs of the protocol participants are given, we can easily determine the set of compatible messages:

$$M_\pi = M_1 \cup M_2 \cup M_3$$

where

$$\begin{aligned} M_1 = \{(\delta, \varepsilon, \omega, \sigma) : \ & \delta = dsc_A, \\ & vfy(k_A,\ (\delta, \varepsilon, \omega),\ \sigma) = \text{true}\} \end{aligned}$$

$$\begin{aligned} M_2 = \{(\gamma, \mu, \sigma) : \ & \mu \in M_1, \\ & fit(\gamma, dsc_B) = \text{true}, \\ & vfy(k_B,\ (\gamma, \mu),\ \sigma) = \text{true}\} \end{aligned}$$

$$\begin{aligned} M_3 = \{(\kappa, \gamma, \delta, \varepsilon, \omega, \sigma, \sigma', \sigma'') : \ & (\gamma, \delta, \varepsilon, \omega, \sigma, \sigma') \in M_2, \\ & fit(dec(\kappa, \varepsilon), dsc_A) = \text{true}, \\ & vfy(k_A,\ (\kappa, \gamma, \delta, \varepsilon, \omega, \sigma, \sigma'),\ \sigma'') = \text{true}\} \end{aligned}$$

5.3 The protocol game

Once the set M_π of compatible messages is determined, we can construct the protocol game G_π of the protocol by applying the framework of Section 3. The player set of the protocol game is $P = \{A, B, net\}$, where A and B represent the main parties, and net represents the network via which the protocol participants communicate with each other. We assume that the network is reliable. The information partition of each player $i \in P$ is determined by i's local state $\Sigma_i(q)$. In order to determine the available actions of the players in $P' = P \setminus \{net\}$, we must tag each message $m \in M_\pi$ with a vector $(\phi_i^m(\Sigma_i(q)))_{i \in P'}$ of logical formulae, where each formula $\phi_i^m(\Sigma_i(q))$ describes the condition that must be satisfied in order for i to be able to send message m in the information set represented by the local state $\Sigma_i(q)$. For the Syverson protocol, these vectors of logical formulae are the following:

- Since B cannot generate valid digital signatures of A, B can send a message $m \in M_1$ only if she received m or a message that contained m earlier. In addition, we assume that A cannot generate a fake item, different from $item_A$, that matches the description dsc_A of $item_A$. Similarly, we assume that A cannot randomly generate a ciphertext ε, and a key κ or a commitment $\omega = w(\kappa)$

$$\begin{aligned}
\varphi_1(\tilde{x},\tilde{m},\tilde{q}) &= ((\exists r < r_{\tilde{x}}(\tilde{q}) : (\textsf{rcv}(\tilde{m}), r) \in H_{\tilde{x}}(\tilde{q})) \vee \\
&\quad (\exists r < r_{\tilde{x}}(\tilde{q}), m' = (\gamma', \tilde{m}, \sigma') \in M_2 : (\textsf{rcv}(m'), r) \in H_{\tilde{x}}(\tilde{q})) \vee \\
&\quad (\exists r < r_{\tilde{x}}(\tilde{q}), m' = (\kappa', \gamma', \tilde{m}, \sigma', \sigma'') \in M_3 : (\textsf{rcv}(m'), r) \in H_{\tilde{x}}(\tilde{q})))
\end{aligned}$$

$$\begin{aligned}
\varphi_2(\tilde{x},\tilde{m},\tilde{q}) &= ((\exists r < r_{\tilde{x}}(\tilde{q}) : (\textsf{rcv}(\tilde{m}), r) \in H_{\tilde{x}}(\tilde{q})) \vee \\
&\quad (\exists r < r_{\tilde{x}}(\tilde{q}), m' = (\kappa', \tilde{m}, \sigma') \in M_3 : (\textsf{rcv}(m'), r) \in H_{\tilde{x}}(\tilde{q})))
\end{aligned}$$

$$\varphi_3(\tilde{x},\tilde{m},\tilde{q}) = (\exists r < r_{\tilde{x}}(\tilde{q}) : (\textsf{rcv}(\tilde{m}), r) \in H_{\tilde{x}}(\tilde{q}))$$

$$\begin{aligned}
\varphi'(\tilde{\gamma},\tilde{q}) &= ((\exists r < r_B(\tilde{q}), m' = (\tilde{\gamma}, \mu', \sigma') \in M_2 : (\textsf{rcv}(m'), r) \in H_B(\tilde{q})) \vee \\
&\quad (\exists r < r_B(\tilde{q}), m' = (\kappa', \tilde{\gamma}, \mu', \sigma', \sigma'') \in M_3 : (\textsf{rcv}(m'), r) \in H_B(\tilde{q})))
\end{aligned}$$

Figure 2: Definition of φ_1, φ_2, φ_3, and φ'

such that $dec(\kappa, \varepsilon)$ matches dsc_A. In other words, if for some message $m = (\delta, \varepsilon, \omega, \sigma) \in M_1$, $fit(dec(w^{-1}(\omega), \varepsilon), dsc_A) = \textsf{true}$ and $dec(w^{-1}(\omega), \varepsilon) \neq item_A$, then A can send m only if she received m or a message that contains m earlier.

Formally, for any $m = (\delta, \varepsilon, \omega, \sigma) \in M_1$:

– if $fit(dec(w^{-1}(\omega), \varepsilon), dsc_A) = \textsf{false}$ or $dec(w^{-1}(\omega), \varepsilon) = item_A$:

$$\begin{aligned}
\phi_A^m(\Sigma_A(q)) &= (\alpha_A(q) = \textsf{true}) \\
\phi_B^m(\Sigma_B(q)) &= (\alpha_B(q) = \textsf{true}) \ \wedge\ \varphi_1(B, m, q)
\end{aligned}$$

– otherwise (i.e., if $fit(dec(w^{-1}(\omega), \varepsilon), dsc_A) = \textsf{true}$ and $dec(w^{-1}(\omega), \varepsilon) \neq item_A$):

$$\begin{aligned}
\phi_A^m(\Sigma_A(q)) &= (\alpha_A(q) = \textsf{true}) \ \wedge\ \varphi_1(A, m, q) \\
\phi_B^m(\Sigma_B(q)) &= (\alpha_B(q) = \textsf{true}) \ \wedge\ \varphi_1(B, m, q)
\end{aligned}$$

where φ_1 is defined in Figure 2.

- Since A cannot generate valid digital signatures of B, A can send a message $m \in M_2$ only if she received m or a message that contains m earlier. For similar reasons, B can send a message $m = (\gamma, \mu, \sigma) \in M_2$ only if she received $\mu \in M_1$ or a message that contains μ earlier. In addition, we assume that B cannot generate a fake item, different from $item_B$, that matches the description dsc_B of $item_B$. This means that if $\gamma \neq item_B$, then B can send m only if she received γ or a message that contains γ earlier.

  Formally, for any $m = (\gamma, \mu, \sigma) \in M_2$:

  – if $\gamma = item_B$:

$$\begin{aligned}
\phi_A^m(\Sigma_A(q)) &= (\alpha_A(q) = \textsf{true}) \ \wedge\ \varphi_2(A, m, q) \\
\phi_B^m(\Sigma_B(q)) &= (\alpha_B(q) = \textsf{true}) \ \wedge\ \varphi_1(B, \mu, q)
\end{aligned}$$

– if $\gamma \neq item_B$:

$$\begin{aligned}\phi_A^m(\Sigma_A(q)) &= (\alpha_A(q) = \mathsf{true}) \wedge \varphi_2(A, m, q) \\ \phi_B^m(\Sigma_B(q)) &= (\alpha_B(q) = \mathsf{true}) \wedge \varphi_1(B, \mu, q) \wedge \varphi'(\gamma, q)\end{aligned}$$

where φ_2 and φ' are defined in Figure 2.

- Since B cannot generate valid digital signatures of A, B can send a message $m \in M_3$ only if she received m earlier (there cannot be another message that contains m in this case). For similar reasons, A can send a message $m = (\kappa, \mu, \sigma) \in M_3$ only if she received $\mu \in M_2$ or a message that contains μ earlier. Note, however, that in general, receiving μ is not sufficient for A to be able to send $m = (\kappa, \mu, \sigma)$, because if the ciphertext ε within μ was not computed by A using the key κ (e.g., if A generated ε randomly), then A may not be able to guess κ. Nevertheless, since our proofs will rely only on the fact that A must receive μ before sending $m = (\kappa, \mu, \sigma)$, we generously give A the power to guess κ, and we consider that receiving μ is also sufficient for A to be able to send $m = (\kappa, \mu, \sigma)$.

  Formally, for any $m = (\kappa, \mu, \sigma) \in M_3$:

$$\begin{aligned}\phi_A^m(\Sigma_A(q)) &= (\alpha_A(q) = \mathsf{true}) \wedge \varphi_2(A, \mu, q) \\ \phi_B^m(\Sigma_B(q)) &= (\alpha_B(q) = \mathsf{true}) \wedge \varphi_3(B, m, q)\end{aligned}$$

  where φ_3 is defined in Figure 2.

The above logical formulae allow us to complete the construction of the protocol game. Before determining the payoffs and describing the strategies that correspond to the programs of the protocol participants, we can already make a few simple statements:

Lemma 1 *If $(\mathsf{snd}(m, B), r) \in H_A(q)$ for some message $m = (\kappa, \mu, \sigma) \in M_3$, round number $r \in \mathbb{N}$, and action sequence $q \in Q$, then there exists $r' < r$ such that $(\mathsf{rcv}(\mu), r') \in H_A(q)$.*

Lemma 2 *If $(\mathsf{snd}(m, A), r) \in H_B(q)$ for some message $m = (\gamma, \mu, \sigma) \in M_2$, round number $r \in \mathbb{N}$, and action sequence $q \in Q$, then there exists $r' < r$ such that $(\mathsf{rcv}(\mu), r') \in H_B(q)$.*

Lemma 3 *Let m be a message in M_3. There is no round number $r < 3$ and action sequence $q \in Q$ such that $(\mathsf{rcv}(m), r) \in H_B(q)$.*

Lemma 4 *Let $m = (\delta, \varepsilon, \omega, \sigma)$ be a message in M_1 such that $fit(dec(w^{-1}(\omega), \varepsilon), dsc_A) = \mathsf{true}$ and $dec(w^{-1}(\omega), \varepsilon) \neq item_A$. There is no player $i \in P'$, round number $r \in \mathbb{N}$, and action sequence $q \in Q$ such that $(\mathsf{rcv}(m), r) \in H_i(q)$.*

Lemma 5 *Let $m = (\gamma, \mu, \sigma)$ be a message in M_2 such that $\gamma \neq item_B$. There is no player $i \in P'$, round number $r \in \mathbb{N}$, and action sequence $q \in Q$ such that $(\mathsf{rcv}(m), r) \in H_i(q)$.*

Lemma 1 states that if A sends a message $m = (\kappa, \mu, \sigma) \in M_3$ in round r in q, then she must receive μ in an earlier round $r' < r$ in q. Similarly, Lemma 2 states that if B sends a message $m = (\gamma, \mu, \sigma) \in M_2$ in round r in q, then she must receive μ in an earlier round $r' < r$ in q. Lemma 3 is a corollary of the first two lemmas that states that B cannot receive a message $m \in M_3$ before round 3. Finally, Lemma 4 states that no player can ever receive a message $m = (\delta, \varepsilon, \omega, \sigma) \in M_1$ such that $fit(dec(w^{-1}(\omega), \varepsilon), dsc_A) = \mathsf{true}$ and $dec(w^{-1}(\omega), \varepsilon) \neq item_A$, and Lemma 5 states that no player can ever receive a message $m = (\gamma, \mu, \sigma) \in M_2$ such that $\gamma \neq item_B$. The proofs of these lemmas are rather straightforward, and can be found in the Appendix.

5.4 Strategies

Based on the programs of the protocol participants described in Subsection 5.2, we can construct the strategies that correspond to the correct behavior of the parties:

Strategy s^*_A

- If $\alpha_A(q) = \textsf{true}$ and $r_A(q) = 1$, then perform the action $\textsf{send}_A(\{(m_1,\ B)\})$, where m_1 is as defined in Figure 1.
- If $\alpha_A(q) = \textsf{true}$ and $r_A(q) = 2$, then perform the action $\textsf{idle}_A$.
- If $\alpha_A(q) = \textsf{true}$ and $r_A(q) = 3$, then let M be the set of those messages $m = (\gamma, \mu, \sigma) \in M_2$ for which $\mu = m_1$ and there exists a round number $r < 3$ such that $(\textsf{rcv}(m), r) \in H_A(q)$. Note that because of Lemma 5, either $M = \emptyset$ or M is a singleton $\{m\}$ where $m = (\gamma, \mu, \sigma) \in M_2$, $\gamma = \mathit{item}_B$, and $\mu = m_1$.
  - If $M = \emptyset$, then perform the action $\textsf{quit}_A$.
  - If $M = \{m\}$, then perform the action $\textsf{send}_A(\{((k, m, \mathit{sig}(k_A^{-1}, (k, m))),\ B)\})$.
- If $\alpha_A(q) = \textsf{true}$ and $r_A(q) = 4$, then perform the action $\textsf{quit}_A$.

Strategy s^*_B

- If $\alpha_B(q) = \textsf{true}$ and $r_B(q) = 1$, then perform the action $\textsf{idle}_B$.
- If $\alpha_B(q) = \textsf{true}$ and $r_B(q) = 2$, then let M be the set of those messages $m \in M_1$ for which there exists a round number $r < 2$ such that $(\textsf{rcv}(m), r) \in H_B(q)$.
  - If $M = \emptyset$, then perform the action $\textsf{quit}_B$.
  - If $M \neq \emptyset$, then choose the smallest message m from M according to some ordering of the messages (e.g., the lexical ordering of bit strings), and perform the action $\textsf{send}_B(\{((\mathit{item}_B, m, \mathit{sig}(k_B^{-1}, (\mathit{item}_B, m))), A)\})$
- If $\alpha_B(q) = \textsf{true}$ and $r_B(q) = 3$, then perform the action $\textsf{idle}_B$.
- If $\alpha_B(q) = \textsf{true}$ and $r_B(q) = 4$, then perform the action $\textsf{quit}_B$.

5.5 Payoffs

We must slightly modify the payoff framework introduced in Subsection 3.7, in order to take into account that the value of item_A diminishes in time[4]. We also have to consider the potential punishment for A if she sends garbage in the first message of the protocol. Taking these into consideration, we define the payoffs of the players as follows.

Let us consider a terminal action sequence q in the protocol game. The payoff of A in q is $y_A(q) = y_A^+(q) - y_A^-(q)$, where $y_A^+(q)$ is the gain and $y_A^-(q)$ is the loss of A in q. Furthermore, the loss of A is defined as $y_A^-(q) = y_A^*(q) + y_A^{**}(q)$, where $y_A^*(q)$ is the loss that stems from losing

[4]We note that time variant values of items lead to time variant payoffs, and the definition of fairness introduced in Section 4 may not be adequate if payoffs are time variant. Since we are not concerned with fairness here, this does not effect our results.

$$\begin{array}{rcl}
\phi_A^+(q) & = & (\exists r \in \mathbb{N}, m = (\gamma, \mu, \sigma) \in M_2 : \\
& & \quad (\gamma = item_B) \ \wedge \ ((\textsf{rcv}(m), r) \in H_A(q))) \\
\\
\phi_A^*(q) & = & (\exists r \in \mathbb{N}, m = (\delta, \varepsilon, \omega, \sigma) \in M_1 : \\
& & \quad (dec(w^{-1}(\omega), \varepsilon) = item_A) \ \wedge \ ((\textsf{snd}(m, B), r) \in H_A(q))) \vee \\
& & (\exists r \in \mathbb{N}, m = (\kappa, \gamma, \delta, \varepsilon, \omega, \sigma, \sigma', \sigma'') \in M_3 : \\
& & \quad (dec(\kappa, \varepsilon) = item_A) \ \wedge \ ((\textsf{snd}(m, B), r) \in H_A(q))) \\
\\
\phi_A^{**}(q) & = & (\exists r \in \mathbb{N}, m = (\delta, \varepsilon, \omega, \sigma) \in M_1 : \\
& & \quad (\mathit{fit}(dec(w^{-1}(\omega), \varepsilon), dsc_A) = \textsf{false}) \ \wedge \ ((\textsf{snd}(m, B), r) \in H_A(q))) \\
\\
\phi_B^+(q, r) & = & (\exists m = (\kappa, \gamma, \delta, \varepsilon, \omega, \sigma, \sigma', \sigma'') \in M_3 : \\
& & \quad (dec(\kappa, \varepsilon) = item_A) \ \wedge \ ((\textsf{rcv}(m), r) \in H_B(q))) \wedge \\
& & (\nexists r' < r, m = (\kappa, \gamma, \delta, \varepsilon, \omega, \sigma, \sigma', \sigma'') \in M_3 : \\
& & \quad (dec(\kappa, \varepsilon) = item_A) \ \wedge \ ((\textsf{rcv}(m), r') \in H_B(q))) \\
\\
\phi_B^-(q) & = & (\exists r \in \mathbb{N}, m = (\gamma, \mu, \sigma) \in M_2 : \\
& & \quad (\gamma = item_B) \ \wedge \ ((\textsf{snd}(m, A), r) \in H_B(q)))
\end{array}$$

Figure 3: Definition of ϕ_A^+, ϕ_A^*, ϕ_A^{**}, ϕ_B^+, and ϕ_B^-

control over $item_A$, and $y_A^{**}(q)$ is the loss that stems from the punishment. The payoff of B in q is $y_B(q) = y_B^+(q) - y_B^-(q)$, where $y_B^+(q)$ is the gain and $y_B^-(q)$ is the loss of B in q.

We denote the values that $item_A$ and $item_B$ are worth to A by u_A^- and u_A^+, respectively. Similarly, we denote the value that $item_B$ is worth to B by u_B^-. The diminishing value of $item_A$ for B is modeled as a function $u_B^+(r)$, which decreases as the round number r increases (see part (a) of Figure 4). We assume that there exists a round number R such that $u_B^+(r) = 0$ for every $r \geq R$, and that breaking a commitment requires more than R rounds. Finally, the value of the punishment is denoted by F. We assume that F is much greater than u_A^+, $u_A^+ > u_A^- > 0$, and $u_B^+(3) > u_B^- > 0$ (see also part (b) of Figure 4).

The gain of A is u_A^+ if A receives a message in M_2 that contains $item_B$, otherwise it is 0. The value of $y_A^*(q)$ is u_A^- if A sends a message in M_1 that contains $item_A$ (in an encrypted form), or if A sends a message in M_3 that contains $item_A$ (in an encrypted form), otherwise it is 0. In addition, the punishment $y_A^{**}(q)$ of A is F if she sends an incorrect message in M_1 that, after breaking the commitment and decrypting the ciphertext in the message, yields an item that does not match the description dsc_A; otherwise the punishment is 0.

The gain of B is $u_B^+(r)$ if B receives a message in M_3 in round r that contains $item_A$ and no such message is received before round r. Note that receiving only a message in M_1 yields no gain for B, because we assume that by the time at which the commitment can be broken, $item_A$ loses its value for B. The loss of B is u_B^- if B sends a message in M_2 that contains $item_B$, otherwise it is 0.

The formal definitions are given below:

$$y_A^+(q) \ = \ \begin{cases} u_A^+ & \text{if } \phi_A^+(q) = \textsf{true} \\ 0 & \text{otherwise} \end{cases}$$

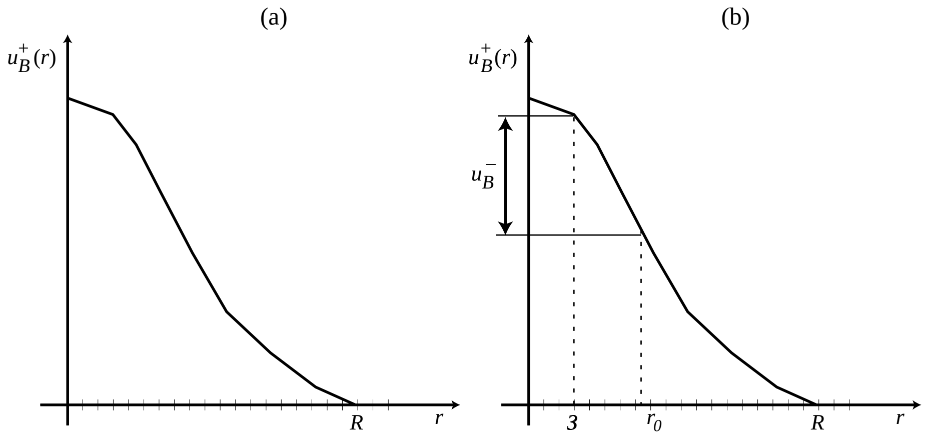

Figure 4: The diminishing value of $item_A$ for B is represented by a decreasing function $u_B^+(r)$. We assume that there exists a round number R such that $u_B^+(r) = 0$ for every $r \geq R$, and that breaking a commitment requires more than R rounds. We also assume that $u_B^+(3) > u_B^- > 0$. Finally, we define r_0 as the smallest round number such that $u_B^+(r_0) \leq u_B^+(3) - u_B^-$.

$$y_A^*(q) = \begin{cases} u_A^- & \text{if } \phi_A^*(q) = \mathsf{true} \\ 0 & \text{otherwise} \end{cases}$$

$$y_A^{**}(q) = \begin{cases} F & \text{if } \phi_A^{**}(q) = \mathsf{true} \\ 0 & \text{otherwise} \end{cases}$$

$$y_B^+(q) = \begin{cases} u_B^+(1) & \text{if } \phi_B^+(q,1) = \mathsf{true} \\ u_B^+(2) & \text{if } \phi_B^+(q,2) = \mathsf{true} \\ \dots & \\ u_B^+(R-1) & \text{if } \phi_B^+(q,R-1) = \mathsf{true} \\ 0 & \text{otherwise} \end{cases}$$

$$y_B^-(q) = \begin{cases} u_B^- & \text{if } \phi_B^-(q) = \mathsf{true} \\ 0 & \text{otherwise} \end{cases}$$

where ϕ_A^+, ϕ_A^*, ϕ_A^{**}, ϕ_B^+, and ϕ_B^- are defined in Figure 3. Note that, by definition, $\phi_B^+(q,r) = \mathsf{true}$ holds for exactly one r, so $y_B^+(q)$ is well defined.

5.6 Proof of rationality

Our proof of rationality relies on the fact that the Syverson protocol is closed for gains and it satisfies the safe back out property:

Lemma 6 *The Syverson protocol is closed for gains.*

Lemma 7 *The Syverson protocol satisfies the safe back out property.*

The proofs of these lemmas are rather straightforward, and can be found in the Appendix.

In order to prove that the Syverson protocol is rational, we have to prove that the strategies s_A^* and s_B^*, which correspond to the correct behavior of the parties, form a Nash equilibrium in the protocol game that we have constructed in Subsections 5.3 and 5.5. In addition, we also have to prove that no other Nash equilibrium is strongly preferable for any of the parties.

Lemma 8 *The strategy profile* $(s^*_{A|\bar{s}}, s^*_{B|\bar{s}})$ *is a Nash equilibrium in the restricted protocol game* $G_{\pi|\bar{s}}$*, where* $\bar{s} = (s^*_{net})$.

Proof: We have to prove that (i) $s^*_{A|\bar{s}}$ is the best response to $s^*_{B|\bar{s}}$, and (ii) $s^*_{B|\bar{s}}$ is the best response to $s^*_{A|\bar{s}}$.

(i) Suppose that there is a strategy $s'_{A|\bar{s}}$ for A such that the payoff of A is higher if she plays $s'_{A|\bar{s}}$ than if she plays $s^*_{A|\bar{s}}$ against $s^*_{B|\bar{s}}$. This means that $y_A(q') > y_A(q^*)$, where $q^* = o_{|\bar{s}}(s^*_{A|\bar{s}}, s^*_{B|\bar{s}})$ and $q' = o_{|\bar{s}}(s'_{A|\bar{s}}, s^*_{B|\bar{s}})$. It is easy to verify that $y_A(q^*) = u^+_A - u^-_A$. Thus, $y_A(q') > y_A(q^*)$ is possible only if $y^+_A(q') = u^+_A$, $y^*_A(q') = 0$, and $y^{**}_A(q') = 0$.

From $y^+_A(q') = u^+_A$, it follows that A received a message $m = (\gamma, \delta, \varepsilon, \omega, \sigma, \sigma') \in M_2$ in q' such that $\gamma = item_B$. This means that B sent m in q'. It follows from Lemma 2 that B can send m only if it received $(\delta, \varepsilon, \omega, \sigma) \in M_1$ from A earlier. Thus, A sent $(\delta, \varepsilon, \omega, \sigma) \in M_1$. Since $y^{**}_A(q') = 0$, $\mathit{fit}(dec(w^{-1}(\omega), \varepsilon), dsc_A)$ must be true. Furthermore, from Lemma 4, we get that $dec(w^{-1}(\omega), \varepsilon) = item_A$. This means that $y^*_A(q')$ cannot be 0.

(ii) Suppose that there is a strategy $s'_{B|\bar{s}}$ for B such that the payoff of B is higher if she plays $s'_{B|\bar{s}}$ than if she plays $s^*_{B|\bar{s}}$ against $s^*_{A|\bar{s}}$. This means that $y_B(q') > y_B(q^*)$, where $q^* = o_{|\bar{s}}(s^*_{A|\bar{s}}, s^*_{B|\bar{s}})$ and $q' = o_{|\bar{s}}(s^*_{A|\bar{s}}, s'_{B|\bar{s}})$. It is easy to verify that $y_B(q^*) = u^+_B(3) - u^-_B$. Let r_0 be the smallest round number such that $u^+_B(r_0) \leq u^+_B(3) - u^-_B$ (see part (b) of Figure 4). Then, $y_B(q') > y_B(q^*)$ is possible only in two cases: (a) $y^+_B(q') = u^+_B(r)$, where $r < r_0$, and $y^-_B(q') = 0$, or (b) $y^+_B(q') = u^+_B(r)$, where $r < 3$. However, case (b) can never occur, because of Lemma 3. Therefore, we have to consider only case (a).

From $y^+_B(q') = u^+_B(r)$, it follows that B received a message $m = (\kappa, \gamma, \delta, \varepsilon, \omega, \sigma, \sigma', \sigma'') \in M_3$ such that $dec(\kappa, \varepsilon) = item_A$ in round r in q'. This means that A sent m in q'. It follows from Lemma 1 that A can send m only if it received $(\gamma, \delta, \varepsilon, \omega, \sigma, \sigma') \in M_2$ from B earlier. Thus, B sent $(\gamma, \delta, \varepsilon, \omega, \sigma, \sigma') \in M_2$. From Lemma 5, we get that $\gamma = item_B$. This means that $y^-_B(q')$ cannot be 0. □

Lemma 9 *Both* A *and* B *prefer* $(s^*_{A|\bar{s}}, s^*_{B|\bar{s}})$ *to any other Nash equilibrium in* $G_{\pi|\bar{s}}$*, where* $\bar{s} = (s^*_{net})$.

Proof: Let us suppose that there exists a Nash equilibrium $(s'_{A|\bar{s}}, s'_{B|\bar{s}})$ in $G_{\pi|\bar{s}}$ such that $y_A(q') > y_A(q^*) = u^+_A - u^-_A$, where $q' = o_{|\bar{s}}(s'_{A|\bar{s}}, s'_{B|\bar{s}})$ and $q^* = o_{|\bar{s}}(s^*_{A|\bar{s}}, s^*_{B|\bar{s}})$. This is possible only if $y^+_A(q') = u^+_A$ and $y^*_A(q') = y^{**}_A(q') = 0$. Since the protocol is closed for gains, $y^+_A(q') = u^+_A > 0$ implies $y^-_B(q') > 0$, and $y^-_A(q') = 0$ implies $y^+_B(q') = 0$. Therefore, if A follows $s'_{A|\bar{s}}$ and B follows $s'_{B|\bar{s}}$, then B's payoff is $y_B(q') = y^+_B(q') - y^-_B(q') < 0$. Note, however, that because of the safe back out property, if B quits at the beginning of the game without doing anything else, then her payoff cannot be negative, whatever strategy is followed by A. This means that $s'_{B|\bar{s}}$ is not the best response to $s'_{A|\bar{s}}$, and thus, $(s'_{A|\bar{s}}, s'_{B|\bar{s}})$ cannot be a Nash equilibrium.

Now let us suppose that there exists a Nash equilibrium $(s'_{A|\bar{s}}, s'_{B|\bar{s}})$ in $G_{\pi|\bar{s}}$ such that $y_B(q') > y_B(q^*) = u^+_B(3) - u^-_B$. This is possible only in two cases: (a) if $y^+_{p2}(q') = u^+_B(r)$, where $r < r_0$ (see part (b) of Figure 4), and $y^-_B(q') = 0$, or (b) if $y^+_B(q') = u^+_B(r)$, where $r < 3$. However, case (b) can never occur, because of Lemma 3. Case (a) can be proven to be impossible using the same technique as in the first part of this proof. □

From Lemma 8 and Lemma 9, we obtain the following:

Proposition 2 *The Syverson protocol is rational.*

5.7 Towards an asynchronous model

In the previous subsection, we proved that Syverson's exchange protocol [27] is rational. However, the proof has been carried out in a model where the network is assumed to be reliable. What if we relax this assumption and allow an unreliable network (i.e., if we assume that there are no bounds on message delivery delays)?

In order to answer this question, our model should be extended with the notion of unreliable network. This can easily be done by giving choices to the network. More precisely, instead of defining the set of available actions for the network as a singleton $\{\mathsf{deliver}_{net}\}$, which means that at the end of each round the network delivers every message that is in the network buffer, we can define the set of available actions for the network as

$$A_{net}(\Sigma_{net}(q)) = \{\mathsf{deliver}_{net}(M) : M \subseteq M_{net}(q)\}$$

which means that the network can deliver any subset of the messages that are currently in the network buffer. Thus, depending on the strategy followed by the network, some messages would not be delivered immediately, but they could stay in the network buffer for some time, even forever.

Note that giving choices to the network to delay the delivery of some messages as described above leads to a more general but still synchronous model, since each player's local state still contains the same current round number. It is possible to define a fully asynchronous model (see [4]), but we do not need it in the following discussion.

On the other hand, we need to extend the definition of rationality, since we must take into account that now the network has several strategies. An easy way to do this is to allow that the strategy vector $\bar{s}$ with which the protocol game is restricted can contain any possible strategy of the network, and to require that the conditions of rationality are satisfied in *every* possible restricted protocol game $G_{\pi|\bar{s}}$, where $\bar{s} = (s^*_{p_3}, s_{net})$, and s_{net} ranges over all the possible strategies of the network. Note that in order to ensure that each player p_i has a unique faithful strategy $s^*_{p_i}$, we must require that p_i has fixed timeout values that specify how many rounds p_i waits for a given type of message.

Let us examine if the Syverson protocol satisfies this extended definition of rationality. Let us assume that both players follow the strategy that corresponds to the faithful execution of the protocol. Furthermore, in order to guarantee the uniqueness of these faithful strategies, let us assume that each of these strategies uses fixed timeout parameters as described above. Now, the network may follow a strategy in which m_3 is delayed, so that B finally timeouts and quits the protocol. This means that there exists a strategy vector $\bar{s}$, and thus a restricted protocol game $G_{\pi|\bar{s}}$, such that $y^+_B(q^*) = 0$ and $y^-_B(q^*) = u^-_B$ (since m_2 has been sent), where $q^* = o_{|\bar{s}}(s^*_{A|\bar{s}}, s^*_{B|\bar{s}})$. Note that the total payoff of B in q^* is negative, so B would be better off if she did not participate in the exchange at all. In other words, $s^*_{B|\bar{s}}$ is not the best response to $s^*_{A|\bar{s}}$ in $G_{\pi|\bar{s}}$, and so $(s^*_{A|\bar{s}}, s^*_{B|\bar{s}})$ cannot be a Nash equilibrium in $G_{\pi|\bar{s}}$. This means that the protocol is not rational in this extended model.

6 Related work

Formal definitions for fair exchange are given by Gaertner *et al.* in [12, 22]. They adopt the formalism of concurrency theory and define fairness based on safety and liveness properties. Although their proposal certainly has a strong potential, it is somewhat limited to fair exchange, and in particular to the concept of strong and weak fairness[5] as it was defined by Asokan in [1]. They do not attempt to formalize the concept of rational exchange, nor to investigate the relationship between rational exchange and fair exchange.

Kremer and Raskin describe a formal approach to the analysis of non-repudiation protocols (which are strongly related to fair exchange protocols) in [17]. They model non-repudiation 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, and use model checking to verify that the protocol satisfies its specification. The paper does not try to formalize the concept of rational exchange nor to relate it to fair exchange.

In [23], Sandholm proposes a method for managing an exchange between two agents – a supplier and a demander – so that the gains from completing the exchange at any point are larger for both agents than the gains from aborting it. The method consists in splitting the exchange into small chunks in a way that the agents can avoid situations that motivate either of them to defect. Sandholm calls this type of exchange *unenforced exchange* (since it does not rely on enforcement from an external trusted party), and relates it to Nash equilibrium. However, he does not formalize the concept of rational exchange in general (the proposed method can be viewed as a particular rational exchange protocol), nor does he relate his results to fair exchange.

In [3], Asokan *et al.* define a formal security model for fair signature exchange. The model is described in terms of a "game", in which a correctly behaving party A and the trusted third party act in a purely reactive fashion, while the actions of the misbehaving party $B*$ are restricted only by a few rules. $B*$ wins the game if it can obtain the digital signature of A on some message m without A obtaining the digital signature of $B*$ on another message m'. They define fairness to mean that the probability that $B*$ wins the game is negligible (with respect to some security parameter). Although, at first sight, the formal model of Asokan *et al.* might seem to be similar to our approach, in fact, it is completely different. First of all, apart from using the terms *game* and *player*, their approach has little to do with game theory as they do not use the notion of equilibrium. Their model is much more similar to the standard models that are used in the cryptographic literature to prove the security of cryptographic algorithms, where one explicitly states the assumptions made about the power of the adversary and tries to prove that the system cannot be broken without invalidating those assumptions. As opposed to this, we completely abstract away cryptography in our model. While the formal model of Asokan *et al.* is probably the most rigorous model that can be found in the literature regarding fairness, it is somewhat restricted to signature exchange protocols. In addition, it does not seem to be appropriate to capture the notion of rationality, which is not a limitation itself, since it was not the goal of the authors to formalize the concept of rational exchange.

Various other approaches to formal analysis of fair exchange protocols are described in [24, 10, 25], but these papers are only loosely related to our work as they do not use game theory (although the model of [10] could easily be related to a game) and they are concerned with fair exchange instead of rational exchange.

7 Conclusion

We presented a formal model of exchange protocols based on game theory, and gave a formal definition for rational exchange and various other properties of exchange protocols, including fairness, within this model. Our model helped us to better understand rational exchange by relating it to the

[5]Strong and weak fairness have nothing to do with the distinction between fairness and rationality.

well-known concept of Nash equilibrium in games. In addition, it also allowed us to study the relationship between rationality and fairness. More specifically, we obtained a formal justification for the intuition that fairness is a stronger requirement than rationality by proving that fairness implies rationality, but the reverse is not true in general.

We illustrated the use of our model for the analysis of existing rational exchange protocols by providing a thorough analysis of a rational exchange protocol proposed by Syverson. We have proved that the Syverson protocol is rational in our model assuming that communication between the protocol parties is reliable. However, as we have seen, if this assumption is relaxed, then the rationality property is lost.

The most original contribution of the paper is the usage of game theory as a tool for modeling and analyzing security protocols. It shows that game theory can successfully be used for such a task. In fact, we believe that it is the most appropriate tool for modeling rational exchange in particular.

Our work was motivated by the "unusual" requirements that designers of exchange protocols should consider in wireless ad hoc networks. We argued that fair exchange protocols (with and without a trusted third party) may not satisfy these requirements. The question that arises is if the Syverson protocol and rational exchange protocols in general are useful in this context. We can only partially answer this question. On the one hand, the Syverson protocol is rational only under the assumption that the communication channel between the protocol parties is reliable. As such, its possible application in ad hoc networks is limited: it may be used between neighboring nodes where there may be good reasons to assume bounds on the message delivery delays, while it certainly cannot be used between distant nodes where bounds on message delivery delays are unrealistic to assume. This argument applies to any synchronous rational exchange protocol (i.e., any exchange protocol that is rational only under the reliable channel assumption). On the other hand, we note that the existence of asynchronous rational exchange protocols (i.e., exchange protocols that are rational even if the reliable channel assumption is relaxed) is an open question. We encourage researchers to consider this question, and if the response is affirmative, to design asynchronous rational exchange protocols.

8 Acknowledgement

The authors are grateful to the anonymous reviewers for their helpful comments and suggestions.

The work presented in this paper was supported (in part) by the Swiss National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

[1] N. Asokan. *Fairness in Electronic Commerce*. PhD thesis, University of Waterloo, Ontario, Canada, May 1998.

[2] N. Asokan and P. Ginzboorg. Key agreement in ad hoc networks. *Computer Communications*, 23:1627–1637, 2000.

[3] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4), April 2000.

[4] L. Buttyán. *Building Blocks for Secure Services: Authenticated Key Transport and Rational Exchange Protocols*. PhD thesis, Swiss Federal Institute of Technology – Lausanne, 2001.

[5] L. Buttyán and J.-P. Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *Proceedings of the IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Boston, MA, USA, August 2000.

[6] L. Buttyán and J.-P. Hubaux. Rational exchange – a formal model based on game theory. In *Proceedings of the 2nd International Workshop on Electronic Commerce (WELCOM)*, November 2001.

[7] L. Buttyán and J.-P. Hubaux. Report on a working session on security in wireless ad hoc networks. *ACM Mobile Computing and Communications Review*, 6(4), October 2002.

[8] L. Buttyán and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), October 2003.

[9] L. Buttyán, J.-P. Hubaux, and S. Čapkun. A formal analysis of Syverson's rational exchange protocol. In *Proceedings of the IEEE Computer Security Foundations Workshop*, June 2002.

[10] R. Chadha, M. Kanovich, and A. Scedrov. Inductive methods and contract-signing protocols. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 176–185, 2001.

[11] R. Cleve. Controlled gradual disclosure schemes for random bits and their applications. In *Advances in Cryptology – CRYPTO'89*, pages 573–588, 1990.

[12] F. Gaertner, H.-H. Pagnia, and H. Vogt. Approaching a formal definition of fairness in electronic commerce. In *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems (Workshop on Electronic Commerce)*, pages 354–359, October 1999.

[13] J.-P. Hubaux, L. Buttyán, and S. Čapkun. The quest for security in mobile ad hoc networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, October 2001.

[14] J.-P. Hubaux, T. Gross, J.-Y. Le Boudec, and M. Vetterli. Towards self-organized mobile ad hoc networks: The Terminodes Project. *IEEE Communications Magazine*, January 2001.

[15] M. Jakobsson. Ripping coins for a fair exchange. In *Advances in Cryptology – EUROCRYPT'95*, pages 220–230, 1995.

[16] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad hoc networks. In *Proceedings of the 9th International Conference on Network Protocols (ICNP)*, November 2001.

[17] S. Kremer and J.-F. Raskin. Formal verification of non-repudiation protocols – a game approach. In *Proceedings of Formal Methods for Computer Security (FMCS 2000)*, July 2000.

[18] Y.-D. Lin and Y.-C. Shu. Multihop cellular: A new architecture for wireless communications. In *Proceedings of Infocom*, 2000.

[19] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[20] O. Markowitch and Y. Roggeman. Probabilistic non-repudiation without trusted third party. In *Proceedings of the 2nd Conference on Security in Communication Networks*, September 1999.

[21] M. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[22] H.-H. Pagnia and F. Gaertner. On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Department of Computer Science, March 1999.

[23] T. Sandholm. Unenforced e-commerce transactions. *IEEE Internet Computing*, 1(6):47–54, November-December 1997.

[24] S. Schneider. Formal analysis of a non-repudiation protocol. In *Proceedings of the IEEE Computer Security Foundations Workshop*, pages 54–65, 1998.

[25] V. Shmatikov and J. Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science*, 283(2):419–450, June 2002.

[26] F. Stajano. *Security for Ubiquitous Computing*. John Wiley and Sons, 2002.

[27] P. Syverson. Weakly secret bit commitment: Applications to lotteries and fair exchange. In *Proceedings of the IEEE Computer Security Foundations Workshop*, pages 2–13, 1998.

[28] S. Čapkun, L. Buttyán, and J.-P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1), January-March 2003.

[29] S. Čapkun, J.-P. Hubaux, and L. Buttyán. Mobility helps security in ad hoc networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, June 2003.

[30] H. Wu, C. Qios, S. De, and O. Tonguz. Integrated cellular and ad hoc relaying systems: iCAR. *IEEE Journal on Selected Areas in Communications*, 19(10), October 2001.

[31] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, November-December 1999.

Appendix: Proofs

Lemma 1

The lemma states that if A sends a message $m = (\kappa, \mu, \sigma) \in M_3$ in round r in q, then she must receive μ in an earlier round $r' < r$ in q.

Proof: Let us suppose that A sends m in round r in q, but she does not receive μ before round r in q. It can be seen from the formulae with which we tagged the messages in M_3 that A can send m in round r only if she receives μ or a message in M_3 that contains μ in an earlier round. By assumption, A does not receive μ before round r, and thus, A must receive a message in M_3 that contains μ before round r.

Let $M_3(\mu) = \{(\kappa', \mu', \sigma') \in M_3 : \mu' = \mu\}$ (i.e., $M_3(\mu)$ contains those messages in M_3 that contain μ). Let r^* be the earliest round in q in which A receives a message in $M_3(\mu)$, and let this message be m^*. Such r^* and m^* exist because (i) we know that A must receive a message in $M_3(\mu)$ before round r in q, and (ii) round numbers are positive integers. In addition, from (i), we get that $r^* < r$ must hold.

Since the network is reliable, if A receives m^* in round r^*, then B sends m^* in round r^*. However, it can be seen from the formulae with which we tagged the messages in M_3 that this is possible only if B receives m^* in an earlier round $\hat{r} < r^*$. This means that A sends m^* in round $\hat{r}$. Again from the formulae with which we tagged the messages in M_3, it can be seen that A can send $m^* \in M_3(\mu)$ in round $\hat{r}$ only if she receives μ or a message in $M_3(\mu)$ before round $\hat{r}$. By assumption, A cannot receive μ before round $\hat{r} < r^* < r$. Thus, she must receive a message in $M_3(\mu)$ before round $\hat{r}$. But this contradicts the fact that the earliest round in which such a message is received by A is r^*. □

Lemma 2

The lemma states that if B sends a message $m = (\gamma, \mu, \sigma) \in M_2$ in round r in q, then she must receive μ in an earlier round $r' < r$ in q.

Proof: Let us suppose that B sends m in round r in q, but she does not receive μ before round r in q. It can be seen from the formulae with which we tagged the messages in M_2 that B can send m in round r only if she receives μ or a message in M_2 or in M_3 that contains μ in an earlier round. By assumption, B does not receive μ before round r, and thus, B must receive a message in M_2 or in M_3 that contains μ before round r.

Let $M_2(\mu) = \{(\gamma', \mu', \sigma') \in M_2 : \mu' = \mu\}$ and $M_3(\mu) = \{(\kappa', \gamma', \mu', \sigma', \sigma'') \in M_3 : \mu' = \mu\}$ (i.e., $M_2(\mu)$ and $M_3(\mu)$ contain those messages in M_2 and in M_3, respectively, that contain μ). If B does not receive any message in $M_2(\mu)$ before round r in q, then let $r_2^* = r$, otherwise, let r_2^* be the earliest round in q in which B receives a message in $M_2(\mu)$. Similarly, if B does not receive any message in $M_3(\mu)$ before round r in q, then let $r_3^* = r$, otherwise, let r_3^* be the earliest round in q in which B receives a message in $M_3(\mu)$.

Now, we can distinguish two cases: (a) $r_2^* \leq r_3^*$ and (b) $r_3^* < r_2^*$.

Case (a): Recall that B must receive a message in $M_2(\mu)$ or in $M_3(\mu)$ before round r in q. This is not possible if $r = r_2^*$. Thus, $r_2^* < r$ must hold. This also means that B receives a message in $M_2(\mu)$ in round r_2^*. Let us denote this message by m_2^*.

If B receives m_2^* in round r_2^*, then A sends m_2^* in round r_2^*. However, it can be seen from the formulae with which we tagged the messages in M_2 that A can send m_2^* in round r_2^* only if she receives (i) m_2^* or (ii) a message $m_3' = (\kappa', m_2^*, \sigma') \in M_3(\mu)$ in an earlier round $\hat{r} < r_2^*$. We show that neither (i) nor (ii) is possible.

(i) If A receives $m_2^* \in M_2(\mu)$ in round $\hat{r}$, then B sends m_2^* in round $\hat{r}$. It can be seen from the formulae with which we tagged the messages in M_2 that this is possible only if B receives μ or a message in $M_2(\mu)$ or in $M_3(\mu)$ before round $\hat{r}$. By assumption, B does not receive μ before round r. Thus, B must receive a message in $M_2(\mu)$ or in $M_3(\mu)$ before round $\hat{r} < r_2^* \leq r_3^*$. However, because of the definitions of r_2^* and r_3^*, B cannot receive any message in $M_2(\mu)$ before r_2^* and any message in $M_3(\mu)$ before r_3^*.

(ii) If A receives $m_3' = (\kappa', m_2^*, \sigma') \in M_3(\mu)$ in round $\hat{r}$, then B sends m_3' in round $\hat{r}$. It can be seen from the formulae with which we tagged the messages in M_3 that this is possible only if B receives m_3' before round $\hat{r} < r_2^* \leq r_3^*$. However, because of the definition of r_3^*, B cannot receive any message in $M_3(\mu)$ before round r_3^*.

Case (b): If $r_3^* < r_2^*$, then $r_3^* < r$ must also hold (since otherwise r_2^* would be greater than r, which is not possible by definition). This means that B receives a message in $M_3(\mu)$ in round r_3^*. Let this message be $m_3^* = (\kappa^*, \gamma^*, \mu, \sigma^*, \sigma^{**})$. If B receives m_3^* in round r_3^*, then A sends m_3^* in round r_3^*. However, from Lemma 1, we know that A can send m_3^* in round r_3^* only if she receives a message $m_2' = (\gamma^*, \mu, \sigma^*) \in M_2(\mu)$ in an earlier round $\hat{r} < r_3^*$. This means that B sends m_2' in round $\hat{r}$. It can be seen from the formulae with which we tagged the messages in M_2 that this is possible only if B receives μ or a message in $M_2(\mu)$ or in $M_3(\mu)$ before round $\hat{r}$. By assumption, B does not receive μ before round r. Thus, B must receive a message in $M_2(\mu)$ or in $M_3(\mu)$ before round $\hat{r} < r_3^* < r_2^*$. However, because of the definitions of r_2^* and r_3^*, B cannot receive any message in $M_2(\mu)$ before round r_2^* and any message in $M_3(\mu)$ before round r_3^*. □

Lemma 3

The lemma states that B cannot receive a message $m \in M_3$ before round 3.

Proof: Let us assume that B receives $m = (\kappa, \gamma, \mu, \sigma, \sigma')$ in round r, where $r < 3$. This means that A sends m in round r. According to Lemma 1, this is possible only if A receives $m' = (\gamma, \mu, \sigma)$ in an earlier round $r' < r$. Thus, B sends m' in round r'. According to Lemma 2, this is possible only if B receives μ in an earlier round $r'' < r' < r$. But this is impossible, since round numbers are positive integers, and $r < 3$. □

Lemma 4

The lemma states that no player can ever receive a message $m = (\delta, \varepsilon, \omega, \sigma) \in M_1$ such that $\mathit{fit}(\mathit{dec}(w^{-1}(\omega), \varepsilon), \mathit{dsc}_A) = \mathsf{true}$ and $\mathit{dec}(w^{-1}(\omega), \varepsilon) \neq \mathit{item}_A$.

Proof: Let us suppose that there exist a player $i \in P'$, a round number $r \in \mathbb{N}$, and an action sequence $q \in Q$ such that $(\mathsf{rcv}(m), r) \in H_i(q)$. This means that a player j sends m in round r in q. According to the logical formulae with which we tagged the messages in M_1, this is possible only if j receives m or a message in M_2 or in M_3 that contains m before round r – no matter whether j is A or B.

Let $M_2(m) = \{(\gamma', \mu', \sigma') \in M_2 : \mu' = m\}$ and $M_3(m) = \{(\kappa', \gamma', \mu', \sigma', \sigma'') \in M_3 : \mu = m\}$. If no player receives m before round r in q, then let $r_1^* = r$, otherwise let r_1^* be the earliest round in q in which m is received by any of the players. If no player receives any message in $M_2(m)$ before round r in q, then let $r_2^* = r$, otherwise let r_2^* be the earliest round in q in which a message in $M_2(m)$ is received by any of the players. Finally, if no player receives any message in $M_3(m)$ before round r in q, then let $r_3^* = r$, otherwise let r_3^* be the earliest round in q in which a message in $M_3(m)$ is received by any of the players.

Now, we can distinguish three cases: (a) $r_1^* \leq r_2^*, r_3^*$, (b) $r_2^* \leq r_1^*, r_3^*$, and (c) $r_3^* \leq r_1^*, r_2^*$.

Case (a): Recall that j receives m or a message in $M_2(m)$ or in $M_3(m)$ before round r in q. Note that if $r_1^* = r$, then no player receives m or any message in $M_2(m)$ or in $M_3(m)$ before round r in q. Thus, $r_1^* < r$ must hold. This means that a player receives m in round r_1^*. If a player receives m in round r_1^*, then a player must send m in round r_1^*. According to the logical formulae with which we tagged the messages in M_1, this is possible only if that player receives m or a message in $M_2(m)$ or in $M_3(m)$ in an earlier round $\hat{r} < r_1^* \leq r_2^*, r_3^*$. However, because of the definitions of r_1^*, r_2^*, and r_3^*, no player can receive m before round r_1^*, a message in $M_2(m)$ before round r_2^*, and a message in $M_3(m)$ before round r_3^*.

Case (b): Recall that j receives m or a message in $M_2(m)$ or in $M_3(m)$ before round r in q. Note that if $r_2^* = r$, then no player receives m or any message in $M_2(m)$ or in $M_3(m)$ before round r in q. Thus, $r_2^* < r$ must hold. This means that a player receives a message $m' = (\gamma', m, \sigma') \in M_2(m)$ in round r_2^*. If a player receives m' in round r_2^*, then a player must send m' in round r_2^*. According to the logical formulae with which we tagged the messages in M_2, A can send m' in round r_2^* only if she receives $m' \in M_2(m)$ or a message in $M_3(m)$ that contains m' in an earlier round $\hat{r} < r_2^* \leq r_1^*, r_3^*$. Furthermore, B can send m' in round r_2^* only if it receives m or a message in $M_2(m)$ or in $M_3(m)$ in

an earlier round $\tilde{r} < r_2^* \leq r_1^*, r_3^*$. However, because of the definitions of r_1^*, r_2^*, and r_3^*, no player can receive m before round r_1^*, a message in $M_2(m)$ before round r_2^*, and a message in $M_3(m)$ before round r_3^*.
Case (c): Recall that j receives m or a message in $M_2(m)$ or in $M_3(m)$ before round r in q. Note that if $r_3^* = r$, then no player receives m or any message in $M_2(m)$ or in $M_3(m)$ before round r in q. Thus, $r_3^* < r$ must hold. This means that a player receives a message $m' = (\kappa', \gamma', m, \sigma', \sigma'') \in M_3(m)$ in round r_3^*. If a player receives m' in round r_3^*, then a player must send m' in round r_3^*. According to the logical formulae with which we tagged the messages in M_3, A can send m' in round r_3^* only if she receives $(\gamma', m, \sigma') \in M_2(m)$ or a message in $M_3(m)$ that contains (γ', m, σ') in an earlier round $\hat{r} < r_3^* \leq r_2^*$. Furthermore, B can send m' in round r_3^* only if she receives $m' \in M_3(m)$ in an earlier round $\tilde{r} < r_3^*$. However, because of the definitions of r_2^*, and r_3^*, no player can receive a message in $M_2(m)$ before round r_2^*, and a message in $M_3(m)$ before round r_3^*. $\Box$

Lemma 5

The lemma states that no player can ever receive a message $m = (\gamma, \mu, \sigma) \in M_2$ such that $\gamma \neq item_B$.

Proof: Let us suppose that there exist a player $i \in P'$, a round number $r \in \mathbb{N}$, and an action sequence $q \in Q$ such that $(\textsf{rcv}(m), r) \in H_i(q)$. This means that a player j sends m in round r in q. According to the logical formulae with which we tagged the messages in M_2, this is possible only if j receives a message in M_2 or in M_3 that contains γ before round r – no matter whether j is A or B.

Let $M_2(\gamma) = \{(\gamma', \mu', \sigma') \in M_2 : \gamma' = \gamma\}$ and $M_3(\gamma) = \{(\kappa', \gamma', \mu', \sigma', \sigma'') \in M_3 : \gamma' = \gamma\}$. If no player receives any message in $M_2(\gamma)$ before round r in q, then let $r_2^* = r$, otherwise let r_2^* be the earliest round in q in which a message in $M_2(\gamma)$ is received by any of the players. If no player receives any message in $M_3(\gamma)$ before round r in q, then let $r_3^* = r$, otherwise let r_3^* be the earliest round in q in which a message in $M_3(\gamma)$ is received by any of the players.

Now, we can distinguish two cases: (a) $r_2^* \leq r_3^*$, and (b) $r_3^* < r_2^*$.
Case (a): Recall that j receives a message in $M_2(\gamma)$ or in $M_3(\gamma)$ before round r in q. Note that if $r_2^* = r$, then no player receives any message in $M_2(\gamma)$ or in $M_3(\gamma)$ before round r in q. Thus, $r_2^* < r$ must hold. This means that a player receives a message $m' = (\gamma, \mu', \sigma') \in M_2(\gamma)$ in round r_2^*. If a player receives m' in round r_2^*, then a player must send m' in round r_2^*. According to the logical formulae with which we tagged the messages in M_2, A can send m' in round r_2^* only if she receives $m' \in M_2(\gamma)$ or a message in $M_3(\gamma)$ that contains m' in an earlier round $\hat{r} < r_2^* \leq r_3^*$. Furthermore, B can send m' in round r_2^* only if it receives a message in $M_2(\gamma)$ or in $M_3(\gamma)$ in an earlier round $\tilde{r} < r_2^* \leq r_3^*$. However, because of the definitions of r_2^* and r_3^*, no player can receive a message in $M_2(\gamma)$ before round r_2^*, and a message in $M_3(\gamma)$ before round r_3^*.
Case (b): Note that $r_3^* < r$ must hold (since otherwise r_2^* would be greater than r, which is not possible by definition). This means that a player receives a message $m' = (\kappa', \gamma, \mu', \sigma', \sigma'') \in M_3(\gamma)$ in round r_3^*. If a player receives m' in round r_3^*, then a player must send m' in round r_3^*. According to the logical formulae with which we tagged the messages in M_3, A can send m' in round r_3^* only if she receives $(\gamma, \mu', \sigma') \in M_2(\gamma)$ or a message in $M_3(\gamma)$ that contains (γ, μ', σ') in an earlier round $\hat{r} < r_3^* < r_2^*$. Furthermore, B can send m' in round r_3^* only if she receives $m' \in M_3(\gamma)$ in an earlier round $\tilde{r} < r_3^*$. However, because of the definitions of r_2^* and r_3^*, no player can receive a message in $M_2(\gamma)$ before round r_2^*, and a message in $M_3(\gamma)$ before round r_3^*. $\Box$

Lemma 6

The lemma states that the Syverson protocol is closed for gains.

Proof: It is enough to prove that for every terminal action sequence q in $G_{\pi|\bar{s}}$, (i) $\phi_A^+(q)$ implies $\phi_B^-(q)$ and (ii) for every r, $\phi_B^+(q, r)$ implies $\phi_A^*(q)$. Both (i) and (ii) follow from the fact that there are only two players A and B who send messages, which means that if player $i \in \{A, B\}$ receives a message m, then the other player $j \in \{A, B\}$, $j \neq i$ must send m. □

Lemma 7

The lemma states that the Syverson protocol satisfies the safe back out property.

Proof: If A does not send any messages in an action sequence q, then $\phi_A^*(q) = \mathsf{false}$ and $\phi_A^{**}(q) = \mathsf{false}$, and thus $y_A^-(q) = 0$. If B does not send any messages in an action sequence q, then $\phi_B^-(q) = \mathsf{false}$, and thus $y_B^-(q) = 0$. □