A Formal Analysis of Syverson's Rational Exchange Protocol[*]

Levente Buttyán Jean-Pierre Hubaux Srdjan Čapkun
Laboratory of Computer Communications and Applications
Swiss Federal Institute of Technology – Lausanne
EPFL-IC-LCA, CH-1015 Lausanne, Switzerland
{levente.buttyan, jean-pierre.hubaux, srdan.capkun}@epfl.ch

Abstract

In this paper, we provide a formal analysis of a rational exchange protocol proposed by Syverson. A rational exchange protocol guarantees that misbehavior cannot generate benefits, and is therefore discouraged. The analysis is performed using our formal model, which is based on game theory. In this model, rational exchange is defined in terms of a Nash equilibrium.

1. Introduction

In [9], Syverson introduces the concept of rational exchange. Rational exchange appears to be similar to fair exchange, but it provides weaker guarantees: A rational exchange protocol does not ensure that a correctly behaving party cannot suffer any disadvantages, but it does guarantee that a misbehaving party cannot gain any advantages. In other words, rational, self-interested parties have no reason to misbehave and to deviate from the protocol (hence the name rational exchange). Rational exchange protocols are proposed in [5, 8, 9, 1].

We started to study the concept of rational exchange in the context of the Terminodes Project[1] [4]. This project is concerned with the design of fully self-organizing mobile ad-hoc networks. Such networks cannot rely on any fixed and pre-installed infrastructure, and therefore, exchange protocols cannot use a trusted third party. Rational exchange seems to be a promising alternative to fair exchange in this environment, since it provides weaker guarantees, and thus, one expects that it has fewer system requirements than fair exchange has. In particular, rational exchange does not always need a trusted third party [8, 9]. Practically, 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.

In [3], we propose a formal model for rational exchange protocols, which is based on game theory. In this model, an exchange protocol is represented as a set of strategies (one strategy for each party) in a game that is constructed from the protocol description. Rational exchange is formally defined in terms of a Nash equilibrium in the protocol game. We also propose formal definitions for various other properties of exchange protocols, including fairness, and we prove that fairness implies rationality, but not vice versa. This justifies the intuition that rational exchange provides weaker guarantees than fair exchange does.

In this paper, we use our protocol game model for the formal analysis of Syverson's rational exchange protocol proposed in [9]. For this reason, we first introduce the protocol game model and the formal definition of rational exchange within this model in Sections 3 and 4, respectively. We keep the presentation brief, since this material has already been presented in [3]. However, for completeness and for making this paper easier to follow, we preferred not to omit this part. Then, in Section 5, we construct the protocol game of the Syverson protocol and prove that it satisfies the definition of rational exchange assuming that the communication between the protocol parties is reliable. Finally, in Section 6, we show that relaxing this assumption leads to the loss of the rationality property.

2. Preliminaries

Before presenting our formal model of exchange protocols, we need to introduce some basic definitions from game theory [7].

2.1. Extensive games

An *extensive game* is a tuple

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

[*] © 2002 IEEE. In Proceedings of the 15th IEEE Computer Security Foundations Workshop, June 2002.

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

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 in $Q \setminus Z$;
- $\mathcal{I}_i$ is an *information partition* of player $i \in P$, which is a partition of the set $\{q \in Q \setminus Z : p(q) = i\}$ with the property that $A(q) = A(q')$ whenever q and q' are in the same *information set* $I_i \in \mathcal{I}_i$;
- $\preceq_i$ 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 $y_i(q) \leq y_i(q')$.

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

2.2. Strategy

A *strategy of player* i 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.

3. Protocol games

There is a striking similarity between games and the situation that occurs when potentially misbehaving parties execute a given exchange protocol:

- each party has choices at various stages during the interaction with the others (e.g., to quit the protocol or to continue);
- the decisions that the parties make determine the outcome of their interaction;
- in order to achieve the most preferable outcome, a misbehaving party may follow a plan that does not coincide with the faithful execution of the exchange protocol.

Therefore, it appears to be a natural idea to model this situation with a game. We refer to this game as the *protocol game*. In this section, we present a general framework for the construction of protocol games from exchange protocols.

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 run the protocol 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 [6], where the same model is used to study the properties of distributed algorithms in a synchronous network system. It is possible to relax this assumption, and to define protocol games for asynchronous systems, but we must omit the details due to space limitations. The interested reader is referred to [2].

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).

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'.

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 $\mathit{snd}(m, j)$ 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 $\mathit{rcv}(m)$ 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

- $\alpha_i(q) \in \{\mathsf{true}, \mathsf{false}\}$ is a boolean, which is 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, $\alpha_i(\epsilon) = \mathsf{true}$, $H_i(\epsilon) = \emptyset$, 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 $j \neq i$.

Now, let us consider an action sequence q, after which player $i \in P'$ has to move. There are two special actions, called idle_i and 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 $\mathsf{send}_i(M)$, 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

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

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

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

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.
- If a player $i \in P'$ performs the action quit_i, then the activity flag of i is set to false. The state of every other player $j \in P \setminus \{i\}$ remains the same as before.
- If a player $i \in P'$ performs 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.
- 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.

3.6. Order of moves

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. Together with the definition of the

	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

available actions (see previous subsection), the above defined order of moves determines the set of possible action sequences and the player function. For a precise definition, the reader is referred to [2].

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).

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 $\phi^+_i(q) = (\exists r : (\mathsf{rcv}(m), r) \in H_i(q))$, 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

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

Note that in $G_{|\bar{s}_{fix}}$, only the players in P_{free} can have several strategies; the players in P_{fix} are bound to the fixed strategies in $\bar{s}_{fix}$. This means that the outcome of $G_{|\bar{s}_{fix}}$ 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 $i \in P_{free}$ and for any strategy $s_i \in S_i$ of player i, let $s_{i|\bar{s}_{fix}}$ denote the strategy that s_i induces in the restricted game $G_{|\bar{s}_{fix}}$. In addition, let us denote the resulting outcome in $G_{|\bar{s}_{fix}}$ when the players in P_{free} follow the strategies in the strategy profile $(s_{i|\bar{s}_{fix}})_{i \in P_{free}}$ by $o_{|\bar{s}_{fix}}((s_{i|\bar{s}_{fix}})_{i \in P_{free}})$.

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

Besides rationality, 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; nevertheless, we need to define them for technical reasons.

Now, we are ready to present the formal definitions:

Definition 1 (Properties of Exchange Protocols) *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 represents 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 $G_{\pi|\bar{s}}$.*
- ***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 $\textsf{quit}_{p_1}$ 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 $\textsf{quit}_{p_2}$ 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 $q = o_{|\bar{s}}(s^0_{p_1|\bar{s}}, s_{p_2|\bar{s}})$.*

5. Analysis of the Syverson protocol

In this section, we analyze the rational exchange protocol proposed by Syverson in [9] 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 descriptions of $item_A$; and k denotes a randomly chosen secret key. In addition, enc is a symmetric-key encryption 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 [9].

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.

[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.

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

Figure 1. Syverson's rational exchange protocol

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 [9]. 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.

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).

[3]Recall that the commitment can be broken, and so the item can be decrypted in a limited amount of time anyhow.

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 m 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:

```
$\pi_A =$
    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

$\pi_B =$
    1. wait until timeout or
       a message $m = (\delta, \varepsilon, \omega, \sigma)$ arrives such that
          - $\delta = dsc_A$
```

```
        - $vfy(k_A,\ (\delta,\varepsilon,\omega),\ \sigma) = \text{true}$
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

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

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

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

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 represents 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 $\omega = w(\kappa)$ such that $dec(\kappa,\varepsilon)$ matches dsc_A. In other words, if for some message $m = (\delta,\varepsilon,\omega,\sigma) \in M_1$, $fit(dec(w^{-1}(\omega),\varepsilon), dsc_A) =$ 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 $fit(dec(w^{-1}(\omega),\varepsilon), dsc_A) =$ false or $dec(w^{-1}(\omega),\varepsilon) = item_A$:

  $$\phi_A^m(\Sigma_A(q)) = (\alpha_A(q) = \text{true})$$
  $$\phi_B^m(\Sigma_B(q)) = (\alpha_B(q) = \text{true}) \ \wedge\ \varphi_1(B,m,q)$$

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

  $$\phi_A^m(\Sigma_A(q)) = (\alpha_A(q) = \text{true}) \ \wedge\ \varphi_1(A,m,q)$$
  $$\phi_B^m(\Sigma_B(q)) = (\alpha_B(q) = \text{true}) \ \wedge\ \varphi_1(B,m,q)$$

  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 $\gamma = item_B$:

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

  – if $\gamma \neq item_B$:

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

  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,

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

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

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

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

Figure 2.

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

$$\begin{aligned}
\phi^m_A(\Sigma_A(q)) &= (\alpha_A(q) = \textsf{true}) \wedge \varphi_2(A, \mu, q) \\
\phi^m_B(\Sigma_B(q)) &= (\alpha_B(q) = \textsf{true}) \wedge \varphi_3(B, m, q)
\end{aligned}$$

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 $(\textsf{snd}(m, B), r) \in H_A(q)$ 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 $(\textsf{rcv}(\mu), r') \in H_A(q)$.*

Lemma 2 *If $(\textsf{snd}(m, A), r) \in H_B(q)$ 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 $(\textsf{rcv}(\mu), r') \in H_B(q)$.*

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 $(\textsf{rcv}(m), r) \in H_B(q)$.*

Lemma 4 *Let $m = (\delta, \varepsilon, \omega, \sigma)$ be a message in M_1 such that $fit(dec(w^{-1}(\omega), \varepsilon), dsc_A) = \textsf{true}$ 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 $(\textsf{rcv}(m), r) \in H_i(q)$.*

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 $(\textsf{rcv}(m), r) \in H_i(q)$.*

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 $fit(dec(w^{-1}(\omega), \varepsilon), dsc_A) = \textsf{true}$ 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 [2].

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 $\alpha_A(q) = \mathsf{true}$ and $r_A(q) = 2$, then perform the action idle_A.
- If $\alpha_A(q) = \mathsf{true}$ 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 $(\mathit{rcv}(m), r) \in H_A(q)$.
  - If $M = \emptyset$, then perform the action quit_A.
  - 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 $\mathsf{send}_A(\{((k, m, \mathit{sig}(k_A^{-1}, (k, m))),\ B)\})$.
- If $\alpha_A(q) = \mathsf{true}$ and $r_A(q) = 4$, then perform the action quit_A.

5.4.1 Strategy s_B^*

- If $\alpha_B(q) = \mathsf{true}$ and $r_B(q) = 1$, then perform the action idle_B.
- If $\alpha_B(q) = \mathsf{true}$ 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 $(\mathit{rcv}(m), r) \in H_B(q)$.
  - If $M = \emptyset$, then perform the action 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 $\mathsf{send}_B(\{((\mathit{item}_B, m, \mathit{sig}(k_B^{-1}, (\mathit{item}_B, m))), A)\})$
- If $\alpha_B(q) = \mathsf{true}$ and $r_B(q) = 3$, then perform the action idle_B.
- If $\alpha_B(q) = \mathsf{true}$ and $r_B(q) = 4$, then perform the action 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. 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 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:

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

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

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

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

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

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.

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

Figure 3.

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 (Gain closed property) *The Syverson protocol is closed for gains.*

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

The proofs of these lemmas are rather straightforward, and can be found in [2].

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 $y_A^{**}(q') = 0$, $fit(dec(w^{-1}(\omega),\varepsilon), dsc_A)$ must be $\textsf{true}$. 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.

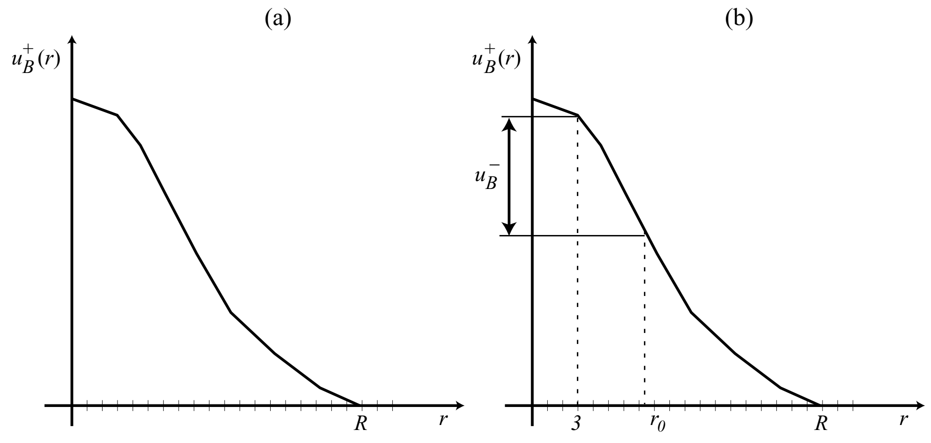

Figure 4. The diminishing value of $item_A$ for B is represented by a decreasing function $u^+_B(r)$. We assume that there exists a round number R such that $u^+_B(r) = 0$ for every $r \geq R$, and that breaking a commitment requires more than R rounds. We also assume that $u^+_B(3) > u^-_B > 0$. Finally,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

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 $G_{\pi|\bar{s}}(L)$ 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 $G_{\pi|\bar{s}}(L)$ such that $y_B(q') > y_B(q^*) = u^+_B(3) - u^-_B$. This is possible only in two cases: (a) if $y^+_{p_2}(q') = u^+_B(r)$, 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 main result of this paper:

Proposition 1 (Rationality) *The Syverson protocol is rational.*

6. Replacing the reliable network with an unreliable one

In the previous section, we proved that Syverson's exchange protocol [9] 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 [2]), 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.

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 correct execution of the protocol. Furthermore, let us assume that each of these strategies has some fixed timeout parameters, which specify the number of rounds that a given player waits for a given message. 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.

7. Conclusion

In this paper, we have reminded the principles of our formal model for exchange protocols, which is based on game theory. By applying this model, we have then provided 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 the communication between the protocol parties is reliable. However, as we have seen, if this assumption is relaxed, then the rationality property is lost.

Our approach makes it possible to examine *any* protocol of similar kind, and to prove that it satisfies (or not) specific properties. As it is quite intuitive, our formalism can be learnt in a matter of days. We believe that this proposal is a significant step towards a systematic method for developing error-free exchange protocols.

References

[1] L. Buttyán. Removing the financial incentive to cheat in micropayment schemes. *IEE Electronics Letters*, 36(2), January 2000.

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

[3] 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.

[4] 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.

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

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

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

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

[9] 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.