Relating Process Algebras and Multiset Rewriting for Security Protocol Analysis

Stefano Bistarelli[1,2] [⋆], Iliano Cervesato[3] [⋆⋆], Gabriele Lenzini[4] [⋆⋆⋆], and Fabio Martinelli[1] [†]

[1] Istituto di Informatica e Telematica–CNR
Via G. Moruzzi, 1 - I-56100 PISA - Italy
{stefano.bistarelli,fabio.martinelli}@iit.cnr.it

[2] Dipartimento di Scienze, Università "D'Annunzio" di Chieti-Pescara
Viale Pindaro 87, 65127 Pescara, Italy
bista@sci.unich.it

[3] Advanced Engineering and Science Division, ITT Industries Inc.
Alexandria, VA 22303 - USA
iliano@itd.nrl.navy.mil

[4] Istituto di Scienza e Tecnologie dell'Informazione–CNR
Via G. Moruzzi, 1 - I-56100 PISA - Italy
lenzini@iei.pi.cnr.it

Abstract. When formalizing security protocols, different specification languages support very different reasoning methodologies, whose results are not directly or easily comparable. Therefore, establishing clear relationships among different frameworks is highly desirable, as it permits various methodologies to cooperate by interpreting theoretical and practical results of one system in another. In this paper, we examine the non-trivial relationship between two general verification frameworks: multiset rewriting (MSR) and a process algebra (PA) inspired to the CCS and the π-calculus. We present two separate mappings, one from MSR to PA and the other from PA to MSR. Although defining a simple and general bijection between MSR and PA appears difficult, we show that in the specific context of cryptographic protocols they do admit effective translations that preserve traces

Keywords: Security Protocols, Multiset Rewriting, Process Algebras.

[⋆] Partially supported by MIUR project "Constraint Based Verification of Reactive Systems" (COVER), and by the MIUR project "Network Aware Programming: Object, Languages, Implementation" (NAPOLI)

[⋆⋆] Partially supported by NRL under contract N00173-00-C-2086.

[⋆⋆⋆] Supported by the MIUR-CNR Project SP4.

[†] Partially supported by MIUR project "Constraint Based Verification of Reactive Systems" (COVER), by MIUR project "MEFISTO", by Microsoft Research and by the CSP project "SeTAPS II"

1 Introduction

In the last decade, security-related problems have attracted the attention of many researchers from several different communities, especially formal methods (*e.g.,* [5, 18, 14, 24, 26, 3, 8, 6, 11, 13, 15, 17, 1]). These researchers have often let their investigation be guided by the techniques and experiences specific to their own areas of knowledge. This massive interest, while on the one hand furthers research, on the other has determined a plethora of different results that often are not directly comparable or integrable with one another. In the last few years, attempts have been made to unify frameworks for specifying security properties usually managed in different ways (*e.g.,* [16]), and to study the relationships between different models for representing security protocols (*e.g.,* [7]).

In this paper we relate uses of process algebras (PA) and multiset-rewriting (MSR) for the description and the analysis of security protocols by defining *encodings* from one formalism to the other. A general comparison between PA and MSR [4] yields results weaker than the specialized application to security protocols analyzed here.

PA is a well-known formal framework and actually denote a family of calculi which have been proposed for describing features of distributed and concurrent systems. Here we use a process algebra that borrows concepts from different calculi, in particular the π-calculus [21] and CCS [20]. It is not difficult to adapt our results to other (value passing) process algebras used in security protocol analysis, *e.g.,* CSP [23] or the *spi*-calculus [2]. In fact, when applied to security protocol analysis, most of them rely only on a well-identified subset of primitives, that have been isolated in the language considered here.

MSR, with its roots in concurrency theory and rewriting logic, has proved to be language of choice for studying foundational issues in security protocols [6]. It is also playing a practical role as the CIL intermediate language [11] of the CASPL security protocol analysis system [10], in particular since translators from several tools to CIL have been developed. For these reasons, MSR has become a central point when comparing languages for protocol specification. In particular, the ties between MSR and strand spaces [25], a popular specification language for crypto-protocols, have been analyzed in [7].

The results of this paper are twofold. First, our encodings establish a firm relationship between the specification methodologies underlying MSR and PA in order to relate verification results obtainable in each model. Second, by bridging PA and MSR, we implicitly define a correspondence between PA and other languages, in particular strand spaces [25] (in a setting with an interleaving semantics), a worthy investigation as remarked in [9]. Interesting work about linear logic and multiset rewriting appears in [19].

The rest of the paper is organized as follows. Section 2 recalls the multiset rewriting and process algebra frameworks and in Section 3 their use in security protocols specification. Section 4 presents the encodings from multiset rewriting to process algebra (Section 4.1), and an inverse (Section 4.2). Section 5 provides the notion of equivalence motivating the encodings. Finally, Section 6 gives some concluding remarks.

2 Background

In this section, we recall the syntax and formal semantics of multiset rewriting (MSR) and of process algebras (PA).

2.1 First Order Multiset Rewriting

The language of first-order MSR is defined by the following grammar:

Elements	$\tilde{a} ::= \cdot \mid a(\boldsymbol{t}), \tilde{a}$
Rewriting Rules	$r ::= \tilde{a}(\boldsymbol{x}) \to \exists \boldsymbol{n}.\tilde{b}(\boldsymbol{x}, \boldsymbol{n})$
Rule sets	$\tilde{r} ::= \cdot \mid r, \tilde{r}$

Multiset elements are chosen as atomic formulas $a(\boldsymbol{t})$ for terms $\boldsymbol{t} = (t_1, \ldots, t_n)$ over some first-order signature Σ. We will write $\tilde{a}(\boldsymbol{x})$ (resp., $t(\boldsymbol{x})$) when we want to emphasize that variables, drawn from $\boldsymbol{x}$, appear in a multiset $\tilde{a}$ (resp., a term t). In the sequel, the comma "," will denote multiset union and will implicitly be considered commutative and associative, while "$\cdot$", the empty multiset, will act as a neutral element (which will allow us to omit it when convenient). The operational semantics of MSR is expressed by the following two judgments:

Single rule application	$\tilde{r} : \tilde{a} \longrightarrow \tilde{b}$
Iterated rule application	$\tilde{r} : \tilde{a} \longrightarrow^* \tilde{b}$

The multisets $\tilde{a}$ and $\tilde{b}$ are called *states* and are always ground formulas. The arrow represents a transition. These judgments are defined as follows:

$$\frac{}{(\tilde{r}, \tilde{a}(\boldsymbol{x}) \to \exists \boldsymbol{n}.\tilde{b}(\boldsymbol{x}, \boldsymbol{n})) : (\tilde{c}, \tilde{a}[\boldsymbol{t}/\boldsymbol{x}]) \longrightarrow (\tilde{c}, \tilde{b}[\boldsymbol{t}/\boldsymbol{x}, \boldsymbol{k}/\boldsymbol{n}])}$$

$$\frac{}{\tilde{r} : \tilde{a} \longrightarrow^* \tilde{a}} \qquad \frac{\tilde{r} : \tilde{a} \longrightarrow \tilde{b} \qquad \tilde{r} : \tilde{b} \longrightarrow^* \tilde{c}}{\tilde{r} : \tilde{a} \longrightarrow^* \tilde{c}}$$

The first inference shows how a rewrite rule $r = \tilde{a}(\boldsymbol{x}) \to \exists \boldsymbol{n}.\tilde{b}(\boldsymbol{x}, \boldsymbol{n})$ is used to transform a state into a successor state: it identifies a ground instance $\tilde{a}(\boldsymbol{t})$ of its antecedent and replaces it with the ground instance $\tilde{b}(\boldsymbol{t}, \boldsymbol{k})$ of its consequent, where $\boldsymbol{k}$ are fresh constants. Here $[\boldsymbol{t}/\boldsymbol{x}]$ denotes the substitution (also written θ) replacing every occurrence of a variable x among $\boldsymbol{x}$ with the corresponding term t in $\boldsymbol{t}$. These rules implement a non-deterministic (in general several rules are applicable at any step) but sequential computation model (one rule at a time). Concurrency is captured as the permutability of (some) rule applications. The remaining rules define $\longrightarrow^*$ as the reflexive and transitive closure of $\longrightarrow$.

2.2 Process Algebras

The language of PA is defined by the following grammar:

Parallel processes	$Q ::= 0 \mid Q \parallel P \mid Q \parallel !P$
Sequential processes	$P ::= 0 \mid a(\boldsymbol{t}).P \mid \overline{a}(\boldsymbol{t}).P \mid [\boldsymbol{t} = \boldsymbol{t}']\, P \mid \nu n.P$

Parallel processes are defined as a parallel composition of, possibly replicated, sequential processes. Sequential processes communicate messages $\boldsymbol{t}$ along channels[1]:an output process $\overline{a}(\boldsymbol{t}).P$ is ready to send a tuple of terms $\boldsymbol{t}$ along the channel a; an input process $a(\boldsymbol{t}).P$ is ready to receive a tuple of messages matching the patterns $\boldsymbol{t}$. The process $[\boldsymbol{t} = \boldsymbol{t'}]\,P$ requires terms $\boldsymbol{t}$ and $\boldsymbol{t'}$ to match in order to continue as P. Finally, the creation of a new object in P (as in the π-calculus [22]) is written as $\nu n.P$. The set $c(t)$ is the constants occurring in a term t is defined as usual, and similary we write $c(Q)$ for the set of constants of a process Q. Free names and free variables are defined as usual. The operational semantics of PA is given by the following judgments:

$$\textit{Single interaction}\ \ Q \Rightarrow Q'; \quad \textit{Iterated interaction}\ \ Q \Rightarrow^* Q'$$

They are defined as follows[2]:

$$\frac{\boldsymbol{t'} = \boldsymbol{t}[\theta]}{(Q \parallel \overline{a}(\boldsymbol{t}).P \parallel a(\boldsymbol{t'}).P') \Rightarrow (Q \parallel P \parallel P'[\theta])}$$

$$\frac{}{(Q \parallel !P) \Rightarrow (Q \parallel !P \parallel P)} \qquad \frac{Q \equiv Q'' \quad Q \Rightarrow Q'}{Q \Rightarrow Q'} \qquad \frac{}{Q \Rightarrow^* Q}$$

$$\frac{c(k) \cap (c(Q) \cup c(P)) = \emptyset}{(Q \parallel \nu n.P) \Rightarrow (Q \parallel P[k/n])} \qquad \frac{\boldsymbol{t} = \boldsymbol{t'}[\theta]}{(Q \parallel [\boldsymbol{t} = \boldsymbol{t'}]\,P) \Rightarrow (Q \parallel P[\theta])} \qquad \frac{Q \Rightarrow Q' \quad Q' \Rightarrow^* Q''}{Q \Rightarrow^* Q''}$$

The first inference (*reaction*) shows how two sequential processes, respectively one ready to perform an output of ground terms $\boldsymbol{t}$, and one ready to perform an input over terms $\boldsymbol{t'}$ react if unification is possible between $\boldsymbol{t}$ and $\boldsymbol{t'}$. The second inference (*replication*), says that $!P$ is essentially a factory of P's. The next inferences describe, respectively, reflexivity of the transition, the generation of a new name, the semantics of match and the transitivity of the transition relation.

3 Security Protocols

We now consider sublanguages of MSR and PA (here referred as MSR$_P$ and PA$_P$) that have gained recent popularity for the specification of cryptographic protocols (see for example [2, 16]). Narrowing our investigation to a specific domain will allow us to directly compare these restricted versions of MSR and PA. The two specifications will rely on a common first-order signature Σ_P. In both formalisms terms in Σ_P stand for messages. Predicate symbols are interpreted as such in MSR$_P$, and as channel names in PA$_P$. Variables will also be allowed in rules and processes.

[1] Here we assume that messages $\boldsymbol{t} = (t_1, \ldots, t_k)$ and channel names a are, resp., as atomic formulas and predicate symbols over some signature Σ.

[2] In the semantics we assume a reasonable structural equivalence relation, $\equiv$, among processes, omitted for saving space.

3.1 Protocols as Multiset Rewriting

MSR$_P$ relies on the following predicate symbols [7]:

Network Messages ($\tilde{N}$): are the predicates used to model the network, where $N(t)$ means that the term t is lying on the network.

Role States ($\tilde{A}$): are the predicates used to model roles. Assuming a set of *role identifiers* $R = \{\rho_1, \ldots, \rho_n\}$, the family of *role state predicates* $\{A_{\rho_i}(\boldsymbol{t}) : i = 0 \ldots l_\rho\}$, is intended to hold the internal state, $\boldsymbol{t}$, of a principal in role $\rho \in R$ during the sequence of protocol steps. The behavior of each role ρ is described through a finite number of rules, indexed from 0 to l_ρ.

Intruder ($\tilde{I}$): are the predicates used to model the intruder I, where $I(t)$, means that the intruder knows the message t.

Persistent Predicates ($\tilde{\pi}$): are ground predicates holding data that does not change during the unfolding of the protocol (*e.g.,* $\mathtt{Kp}(K, K')$ indicates that K and K' form a pair of public/private keys). Rules use these predicates to access the value of persistent data.

A security protocol is expressed in MSR$_P$ as a set of rewrite rules $\tilde{r}$ of a specific format called a *security protocol theory.* Given roles R, it can be partitioned as $\tilde{r} = \cup_{\rho \in R}(\tilde{r}_\rho), \tilde{r}_I$, where $\tilde{r}_\rho$ and $\tilde{r}_I$ describe the behavior of a role $\rho \in R$ and of the intruder I. For each role ρ, the rules in $\tilde{r}_\rho$ consist of:

– one *instantiation rule* $r_{\rho_0} : \tilde{\pi}(\boldsymbol{x})\exists \boldsymbol{n}.A_{\rho_0}(\boldsymbol{n}, \boldsymbol{x}), \tilde{\pi}(\boldsymbol{x})$
– zero or more ($i = 1 \ldots l_\rho$) *message exchange rules*:

$$\begin{array}{llrcl}
\textit{send} & r_{\rho_i} : & A_{\rho_{i-1}}(\boldsymbol{x}) & \rightarrow & A_{\rho_i}(\boldsymbol{x}), N(t(\boldsymbol{x})) \\
\textit{receive} & r_{\rho_i} : & A_{\rho_{i-1}}(\boldsymbol{x}), N(t(\boldsymbol{x}, \boldsymbol{y})) & \rightarrow & A_{\rho_i}(\boldsymbol{x}, \boldsymbol{y})
\end{array}$$

Rules in $\tilde{r}_I$ are the standard rules describing the intruder in the style of Dolev-Yao [12], whose capabilities consist in intercepting, analyzing, synthesizing and constructing messages, with the possibility to access to permanent data. Formally:

$$\begin{array}{lc|lc}
r_{I_1}: & \pi(x) \rightarrow I(x), \pi(x) & r_{I_2}: & \cdot \rightarrow \exists n.I(n) \\
r_{I_3}: & I(x) \rightarrow N(x) & r_{I_4}: & N(x) \rightarrow I(x) \\
r_{I_5}: & I(\langle x_1, x_2 \rangle) \rightarrow I(x_1), I(x_2) & r_{I_6}: & I(x_1), I(x_2) \rightarrow I(\langle x_1, x_2 \rangle) \\
r_{I_7}: & I(\{x\}_k), I(k), \mathtt{Kp}(k, k') \rightarrow I(x), \mathtt{Kp}(k, k') & r_{I_8}: & I(x), I(k) \rightarrow I(\{x\}_k) \\
r_{I_9}: & I(x) \rightarrow \cdot & r_{I_{10}}: & I(x) \rightarrow I(x), I(x)
\end{array}$$

where x and k are variables. In MSR$_P$, a state is a multiset of the form $\tilde{s} = (\tilde{A}, \tilde{N}, \tilde{I}, \tilde{\pi})$, where the components collect ground facts of the form $N(t)$, $A_{\rho_i}(\boldsymbol{t})$, $I(t)$ and $\pi(\boldsymbol{t})$, respectively. An *initial state* $\tilde{s}_0 = (\tilde{\pi}, \tilde{I}_0)$ contains only persistent predicates ($\tilde{\pi}$) and initial intruder knowledge ($\tilde{I}_0$). A pair ($\tilde{r} : \tilde{s}$) consisting of an protocol theory $\tilde{r}$ and a state $\tilde{s}$ is called a *configuration.* The initial configuration is ($\tilde{r} : \tilde{s}_0$).

3.2 Protocols as Processes

A security protocol may be described in a fragment of PA where: (a) every communications happen through the net (here P_{net} is the process that manages the net as a public channel where each P_ρ sends and receives messages); (b) there is an intruder, with some initial knowledge able to intercept and forge messages passing through the net (here $Q_{!I}$, with initial knowledge Q_{I_0}); each principal starts a protocol interpreting a certain role ρ.

A security protocol, involving a collection of roles ρ, is expressed in PA$_P$ as a *security protocol process* Q_0, defined by as the parallel composition of five components: $P_{net} \parallel \prod_\rho !P_\rho \parallel Q_{!I} \parallel Q_{I_0} \parallel Q_{!\pi}$ where $\prod \mathcal{P}$ denotes the parallel composition of all the processes in $\mathcal{P}$. Precisely:

$P_{net} = !(N_i(x).\overline{N_o}(x).0)$**.** describes the behavior of the network. It simply copies messages from channel N_i (input of the net) to N_o (output of the net), implementing an asynchronous form of message transmission.

P_ρ**.** Each of these processes represents the sequence of actions that constitute a role, in the sense defined for MSR$_P$. These processes have the following form[3]: $P_\rho = \tilde{\pi}(\boldsymbol{x}).\nu\boldsymbol{n}.P'_\rho$ where P'_ρ is a sequential process that does input and output only on net's channels or does some terms matching. Formally, $P'_\rho ::= 0 \;\mid\; N_o(\boldsymbol{t}).P'_\rho \;\mid\; \overline{N_i}(\boldsymbol{t}).P'_\rho \;\mid\; [\boldsymbol{t} = \boldsymbol{t'}]\, P'_\rho$.

$Q_{!I} = !P_{I_1} \parallel \ldots \parallel !P_{I_{10}}$. This is the specification of the intruder model in a Dolev-Yao style. Each P_{I_i} describes one capability of the intruder. The dedicated channel I holds the information the intruder operates on (it can be either initial, intercepted, or forged). They are defined as follows:

$$\begin{array}{l|l}
P_{I_1} = \pi(x).\overline{I}(x).0 & P_{I_2} = \nu n.\overline{I}(n).0 \\
P_{I_3} = N_o(x).\overline{I}(x).0 & P_{I_4} = I(x).\overline{N_i}(x).0 \\
P_{I_5} = I(\langle x_1,\, x_2\rangle).\overline{I}(x_1).\overline{I}(x_2).0 & P_{I_6} = I(x_1).I(x_2).\overline{I}(\langle x_1,\, x_2\rangle).0 \\
P_{I_7} = I(\{y\}_k).I(k).\mathtt{Kp}(\langle k,\, k'\rangle).\overline{I}(y).0 & P_{I_8} = I(x).I(k).\overline{I}(\{x\}_k).0 \\
P_{I_9} = I(x).0 & P_{I_{10}} = I(x).\overline{I}(x).\overline{I}(x).0
\end{array}$$

$Q_{I_0} = \prod \overline{I}(t).0$ for some terms t. Q_{I_0} represents the initial knowledge of the intruder.

$Q_{!\pi} = \prod !\overline{\pi}(t).0$ represents what we called "persistent information" in the case of MSR$_P$. We can assume the same predicate (here channel) names with the same meaning. This information is made available to client processes on each channel π (*e.g.,* Kp). It is assumed that no other process performs an output on π.

In PA$_P$, a *state* is a process of the form $Net \parallel \prod_\rho Role_\rho \parallel Eve \parallel Init \parallel Q'$ where Net, $Role_\rho$, Eve and $Init$ are ground, un-replicated (*i.e.,* without a bang as a prefix) instances of sequential suffixes of processes P_{net}, $\prod_\rho !P_\rho$, $(Q_{!I} \parallel Q_{I_0})$ and $Q_{!\pi}$ respectively, while $Q' = (P_{net} \parallel \prod_\rho !P_\rho \parallel Q_{!I} \parallel Q_{!\pi})$ collects the parts of the specification subject to replication.

[3] Here we use $\tilde{\pi}(\boldsymbol{x}).P$ as a shortcut for $\pi_1(\boldsymbol{t}_1(\boldsymbol{x}_1))\ldots\pi_k(\boldsymbol{t}_k(\boldsymbol{x}_k)).P$, and $\nu\boldsymbol{n}.P$ for $\nu n_1.\ldots.\nu n_h.P$.

4 Encodings for Protocol Specifications

This section describes the encodings from MSR$_P$ to PA$_P$ and vice versa. As above, we assume the same underlying signature Σ_P. In particular, the predicate symbols and terms in MSR$_P$ find their counterpart in channel names and messages in PA$_P$, respectively.

The first mapping, from MSR$_P$ to PA$_P$, is based on the observation that role state predicates force MSR$_P$ rules to be applied sequentially within a role (this is not true for general MSR theories [4]). Minor technicalities are involved in dealing with terms and with the presence of multiple instances of a same role (they are addressed by unification and banged processes, respectively). At its core, the inverse encoding, from PA$_P$ to MSR$_P$, maps sequential agents to a set of MSR$_P$ rules corresponding to roles: we generate appropriate role state predicates in correspondence of the intermediate stages of each sequential process. The bang operator is not directly involved in this mapping as it finds its counterpart in the way rewriting rules are applied. The transformation of the intruder, whose behavior is fixed a priori, is treated off-line in both directions. 0

4.1 From MSR$_P$ to PA$_P$

Given an initial MSR$_P$ configuration $(\cup_\rho(\tilde{r}_\rho), \tilde{r}_I : \tilde{\pi}, \tilde{I}_0)$, we return an PA$_P$ security protocol process $P_{net} \parallel \prod_\rho !P_\rho \parallel Q_{!I} \parallel Q_{I_0} \parallel Q_{!\pi}$. In particular (a) P_{net} is fixed a priori (see Section 3.2); (b) $\prod_\rho !P_\rho$ and $Q_{!I}$, result from the transformation of respectively $\cup_\rho(\tilde{r}_\rho)$ and $\tilde{r}_I$; (c) $Q_{!\pi}$ and Q_{I_0} result from transformation of $\tilde{\pi}, \tilde{I}_0$.

Processes P_ρ, for each role ρ, are obtained via the transformation function $\lceil \_ \rceil$ ranging over the set of role rules $\cup_\rho(\tilde{r}_\rho)$. We define it depending on the structure of the role rule $r_{\rho_i} \in \tilde{r}_\rho$ involved. Formally for $i = 0$:

$$\lceil r_{\rho_0} \rceil = \tilde{\pi}(\boldsymbol{x}).\nu \boldsymbol{n}.\lceil r_{\rho_1} \rceil \qquad \text{if } r_{\rho_0} : \tilde{\pi}(\boldsymbol{x}) \rightarrow \exists \boldsymbol{n}.A_{\rho_0}(\boldsymbol{x}), \tilde{\pi}(\boldsymbol{x})$$

Informally role generation rules are mapped onto a process which first receives, in sequence, permanent terms via the channels π in $\tilde{\pi}$ and then generates all the new names $\boldsymbol{n}$ used along the execution of the protocol. For $0 < i \leq l_\rho - 1$:

$$\lceil r_{\rho_{i+1}} \rceil = \begin{cases} [\boldsymbol{t} = \boldsymbol{t''}]\, \overline{N_i}(t).\lceil r_{\rho_{i+2}} \rceil, \text{ if } r_{\rho_{i+1}} = A_{\rho_i}(\boldsymbol{x}) \rightarrow A_{\rho_{i+1}}(\boldsymbol{x}), N(t) \\ [\boldsymbol{t} = \boldsymbol{t''}]\, N_o(t)\lceil r_{\rho_{i+2}} \rceil, \text{ if } r_{\rho_{i+1}} = A_{\rho_i}(\boldsymbol{x}), N(t) \rightarrow A_{\rho_{i+1}}(\boldsymbol{t'}) \end{cases}$$

where $\boldsymbol{t''}$ are the terms appearing in the *consequent* $A_{\rho_i}(\boldsymbol{t''})$ of the rule r_{ρ_i}. Here we exploit the fact that we know in advance the next rule to be applied. In this way we can produce the matching primitive following the receive that is sound with the next rule selection. Finally we have, with a little abuse of notation, $\lceil r_{\rho_{l_\rho+1}} \rceil = 0$. The final process defining the role ρ behavior is the following: $P_\rho \overset{def}{=} \lceil r_{\rho_0} \rceil$

The intruder is handled by simply mapping $\tilde{r}_I$ to $Q_{!I}$. More precisely, we define the transformation function $\lceil \_ \rceil_I$ such that relates the intruder rewriting rule r_{I_j} with the sequential agents P_{I_j} (see Section 3.2). Being the intruder behavior fixed *a priori*, this transformation is effectively performed off-line once and for all.

At this point the transformation is complete as soon as $\tilde{s}_0 = \tilde{\pi}, \tilde{I}_0$ is treated. The multiset $\tilde{I}_0$ guides the definition of the process Q_{I_0}, while the multiset $\tilde{\pi}$ guides the definition of the process $Q_{!\pi}$, that is: $Q_{I_0} \overset{def}{=} \prod_{I(t)\in\tilde{I}_0}(\overline{I}(t).0)$, $Q_{!\pi} \overset{def}{=} \prod_{\pi(t)\in\tilde{\pi}}(!\overline{\pi}(t).0)$. Writing, with a little abuse of notation, $\lceil \_ \rceil$ the generic function that, given a multiset rewriting MSR$_P$ security protocol theory returns an PA$_P$ security protocol process, the encoding can be summarized as:

$$P_{net} \parallel \prod_{\rho} !P_\rho \parallel Q_{!I} \parallel Q_{I_0} \parallel Q_{!\pi} = \lceil (\cup_\rho(\tilde{r}_\rho), \tilde{r}_I : \tilde{\pi}, \tilde{I}_0) \rceil$$

4.2 From PA$_P$ to MSR$_P$

Given a PA$_P$ security protocol process $P_{net} \parallel \prod_\rho !P_\rho \parallel Q_{!I} \parallel Q_{I_0} \parallel Q_{!\pi}$ we show how to construct an initial state in MSR$_P$. The basic translation involves the transformation function $\lfloor \_ \rfloor^{\#}_{(i;\epsilon)}$ (called as a subroutine by the top level transformation $\lfloor \_ \rfloor$) which, given a sequential agent representing a role ρ, returns the multiset of rules $\tilde{r}_\rho$. Here i is an non negative integer. Formally:

$$\begin{array}{lcl}
\lfloor \tilde{\pi}(\boldsymbol{x}).\nu\boldsymbol{n}.P'_\rho \rfloor & = & \{\tilde{\pi}(\boldsymbol{x}) \to \exists \boldsymbol{n}.A_{\rho 0}(\boldsymbol{n},\boldsymbol{x})\} \;\cup\; \lfloor P'_\rho \rfloor^{\#}_{(1;(\boldsymbol{x},\boldsymbol{n}))} \\
\lfloor N_o(t(\boldsymbol{y})).P'_\rho \rfloor^{\#}_{(i;\boldsymbol{x})} & = & \{A_{\rho_i - 1}(\boldsymbol{x}), N(t(\boldsymbol{y})) \to A_{\rho_i}(\boldsymbol{x},\boldsymbol{y})\} \;\cup\; \lfloor P'_\rho \rfloor^{\#}_{(i+1;(\boldsymbol{x},\boldsymbol{y}))} \\
\lfloor \overline{N_i}(t(\boldsymbol{y})).P''_\rho \rfloor^{\#}_{(i;\boldsymbol{x})} & = & \{A_{\rho_i - 1}(\boldsymbol{x}) \to A_{\rho i}(\boldsymbol{x}), N(t(\boldsymbol{y}))\} \;\cup\; \lfloor P'_\rho \rfloor^{\#}_{(i+1;\boldsymbol{x})} \\
\lfloor [\boldsymbol{t} = \boldsymbol{t'}]\; P'_\rho \rfloor^{\#}_{(i;\boldsymbol{x})} & = & \lfloor P'_\rho[\theta] \rfloor^{\#}_{(i;\boldsymbol{x}[\theta])} \quad \text{where } \boldsymbol{t} = \boldsymbol{t'}[\theta] \\
\lfloor 0 \rfloor^{\#}_{(i;\boldsymbol{x})} & = & \cdot
\end{array}$$

The intruder is handled by the inverse of the transformation $\lceil \_ \rceil_I$, which we call $\lfloor \_ \rfloor_I$. The initial state $\tilde{s}_0 = \tilde{\pi}, \tilde{I}_0$ is defined so that $\tilde{\pi} = \{\pi(\boldsymbol{t}) : \overline{\pi}(\boldsymbol{t}) \in P_{init}\}$ and $\tilde{I}_0 = \{I(t) : \overline{I}(t) \in Q_{I_0}\}$,[4] that is $\tilde{s}_0$ is the multiset of persistent facts which are isomorphic to the predicates appearing in process P_{init}, and the multiset of intruder predicates corresponding of messages that appear in Q_{I_0}. Writing, with a little abuse of notation, $\lfloor \_ \rfloor$ the generic function that given a PA$_P$ security protocol process returns a MSR$_P$ security protocol theory, the whole transformation can be summarized as:

$$(\cup_\rho(\tilde{r}_\rho), \tilde{r}_I : \tilde{\pi}, \tilde{I}_0) = \lfloor P_{net} \parallel \prod_{\rho} !P_\rho \parallel Q_{!I} \parallel Q_{I_0} \parallel Q_{!\pi} \rfloor$$

5 Correspondence

In this section, we will call an MSR$_P$ configuration and a PA$_P$ state *corresponding* when they manifest the same network and intruder behavior, step by step. We first formalize the notion of transition step in each formalism.

[4] A formal definition of $a(\boldsymbol{t}) \in Q$ is given in the next Section.

Definition 1. *Given a process Q, $Q \xrightarrow{\alpha}$ says that α is the multiset of communication events that the process Q may perform in a next step of execution. Formally:*

$$\frac{}{0 \xrightarrow{\cdot}} \qquad \frac{Q \xrightarrow{\alpha} \qquad P \xrightarrow{\alpha'}}{(Q \parallel P) \xrightarrow{\alpha,\alpha'}} \qquad \frac{}{\nu n.P \xrightarrow{\cdot}}$$

$$\frac{}{a(\boldsymbol{t}).P \xrightarrow{\{a(\boldsymbol{t})\}}} \qquad \frac{}{\overline{a}(\boldsymbol{t}).P \xrightarrow{\{\overline{a}(\boldsymbol{t})\}}} \qquad \frac{P \xrightarrow{\alpha} \qquad \boldsymbol{t} = \boldsymbol{t'}[\theta]}{([\boldsymbol{t} = \boldsymbol{t'}]\ P) \xrightarrow{\alpha[\theta]}} \qquad \frac{P \xrightarrow{\alpha'}}{!P \xrightarrow{\alpha,\alpha'}}$$

We write (overloading the include symbol) $\alpha' \in P$ if $\alpha' \in \{\alpha : P \xrightarrow{\alpha}\}$ Let us consider a MSR$_P$ state $(\tilde{r} : \tilde{\pi}, \tilde{A}, \tilde{N}, \tilde{I})$, and a PA$_P$ state $Net \parallel Eve \parallel \prod_\rho Role_\rho \parallel Init \parallel Q'$. We want to define a correspondence between configurations such that, if in MSR$_P$ is possible to perform an action (by applying a rule) then in PA$_P$ is possible to proceed because of some transitions, and vice versa.

Definition 2. *Given a* MSR$_P$ *configuration $C = (\tilde{r} : \tilde{\pi}, \tilde{A}, \tilde{N}, \tilde{I})$. We define: $\mathcal{H}^-_{\tilde{N}}(C) = \{t : N(t) \in \tilde{N}\}$ and $\mathcal{H}^-_{\tilde{I}}(C) = \{t : I(t) \in \tilde{I}\}$ that is, resp., all the messages lying on the net and all the messages known by the intruder in the state in C.*

Definition 3. *Given a channel name c and a* PA$_P$ *state $Q = Net \parallel Eve \parallel \prod_\rho Role_\rho \parallel Init \parallel Q'$, we define: $\mathcal{K}^{\overline{N}}_{Net}(Q) = \{\boldsymbol{t} : \overline{N}(\boldsymbol{t}) \in Net\}$ and $\mathcal{K}^{\overline{I}}_{Eve}(Q) = \{\boldsymbol{t} : \overline{I}(\boldsymbol{t}) \in Eve\}$. that is the messages that processes, resp., Eve and Net may sent in output over channel I and N, in a next transition from state Q.*

Using Definitions 2 and 3, we could make precise what we intend for a MSR$_P$ configuration and a PA$_P$ state to be correspondent.

Definition 4. *Given a* MSR$_P$ *configuration C and a* PA$_P$ *state Q. We say that C and Q are* correspondent, *written as $C \bowtie Q$ if and only if the two conditions hold:*

1. $\mathcal{H}^-_{\tilde{N}}(C) = \mathcal{K}^{\overline{N}}_{Net}(Q)$
2. $\mathcal{H}^-_{\tilde{I}}(C) = \mathcal{K}^{\overline{I}}_{Eve}(Q)$

Informally $C \bowtie Q$ says that the messages that are lying on the net and the intruder knowledge are the same in states C and Q. Now we can define an equivalence relation between a MSR$_P$ and a PA$_P$ states.

Definition 5. *A binary relation*[5] $\sim\ \subseteq \mathcal{C} \times \mathcal{Q}$ *is a* strong correspondence *if $(\tilde{r} : \tilde{s}) \sim Q$ implies that:*

1. $(\tilde{r} : \tilde{s}) \bowtie Q$;
2. $\tilde{r} : \tilde{s} \longrightarrow \tilde{s}'$ *then* $Q \Rightarrow^* Q'$ *and* $(\tilde{r} : \tilde{s}') \sim Q'$;
3. $Q \Rightarrow Q'$ *then* $\tilde{r} : \tilde{s} \longrightarrow^* \tilde{s}'$ *and* $(\tilde{r} : \tilde{s}') \sim Q'$.

[5] $\mathcal{C}$ and $\mathcal{Q}$ are the set of all MSR$_P$ configurations and PA$_P$ states.

The following theorems informally affirm that security protocol specifications among MSR_P and PA_P, obtained via the encodings here presented, are correspondent.

Theorem 1. *Given an* MSR_P *security protocol theory* C_0*. Then* $C_0 \sim \lceil C_0 \rceil$*.*

Theorem 2. *Given an* PA_P *security protocol process* Q_0*. Then* $\lfloor Q_0 \rfloor \sim Q_0$*.*

6 Conclusions

This paper shows how multiset rewriting theories (MSR_P) and processes algebras (PA_P) used to describe security protocols may be related. Indeed we show how to define transformations between MSR_P to PA_P specifications of a security protocol, whose semantics (based on labeled transition systems) are proved to be related. The paper introduces a correspondence relation based on what messages appear on the net and on what messages the intruder knows. A direct consequence of this results is that the given transformations preserve confidentiality properties, *i.e.,* the ability or inability of the intruder to discover a message. Moreover, since several forms of authentication among protocol participants may be formulated in terms of properties on what is sent to the net or what is captured by the intruder, authentication is also preserved by our encodings.

References

[1] M. Abadi and B. Blanchet. Analyzing Security Protocols with Secrecy Types and Logic Programs. *ACM SIGPLAN Notices*, 31(1):33–44, 2002. Proc. of the 29th ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages (POPL'02).

[2] M. Abadi and A. D. Gordon. Reasoning about Cryptographic Protocols in the Spi Calculus. In *Proc. of CONCUR '97: Concurrency Theory, 8th International Conference*, volume 1243 of *Lecture Notes in Computer Science*, pages 59–73. Springer-Verlag, 1997.

[3] M. Abadi and A. D. Gordon. A Bisimulation Methods for Cryptographic Protocols. In *Proc. of ESOP'98*, 1998.

[4] S. Bistarelli, I. Cervesato, G. Lenzini, and F. Martinelli. Relating multiset rewriting and process algebras (for example for security protocol analysis). Technical report, Naval Laboratory Research, 2003. to appear.

[5] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Proc. of the Royal Society of London*, volume 426 of *Lecture Notes in Computer Science*, pages 233–271. Springer-Verlag, 1989.

[6] I. Cervesato, N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. A Meta-Notation for Protocol Analysis. In *Proc. of the 12th IEEE Computer Security Foundations Workshop (CSFW'99).* IEEE Computer Society Press, 1999.

[7] I. Cervesato, N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. Relating strands and multiset rewriting for security protocol analysis. In *Proc. of the 13th IEEE Computer Security Foundations Workshop (CSFW '00)*, pages 35–51. IEEE, 2000.

[8] E. M. Clarke, S. Jha, and W. Marrero. A Machine Checkable Logic of Knowledge for Protocols. In *Proc. of Workshop on Formal Methods and Security Protocols*, 1998.

[9] F. Crazzolara and G. Winskel. Events in security protocols. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 96–105. ACM Press, 2001.

[10] G. Denker and J. Millen. Capsl integrated protocol environment. In *Proc. of DARPA Information Survivability Conference (DISCEX 2000), pp 207-221, IEEE Computer Society, 2000*, 2000.

[11] G. Denker, J. K. Millen, A. Grau, and J. K. Filipe. Optimizing protocol rewrite rules of CIL specifications. In *CSFW*, pages 52–62, 2000.

[12] D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transaction on Information Theory*, 29(2):198–208, 1983.

[13] M. Fiore and M. Abadi. Computing Symbolic Models for Verifying Cryptographic Protocols. In *Proc. of the 14th Computer Security Foundation Workshop (CSFW-14)*, pages 160–173. IEEE, Computer Society Press, 2001.

[14] R. Focardi and R. Gorrieri. The Compositional Security Checker: A tool for the Verification of Information Flow Security Properties. *IEEE Transactions on Software Engineering*, 23(9):550–571, 1997.

[15] R. Focardi, R. Gorrieri, and F. Martinelli. NonInterference for the Analysis of Cryyptographic Protocols. In *Proc. of the ICALP'00.* Springer-Verlag, 2000.

[16] R. Focardi and F. Martinelli. A Uniform Approch for the Definition of Security Properties. In *Proc. of Congress on Formal Methods (FM'99)*, volume 1708 of *Lecture Notes in Computer Science*, pages 794–813. Springer-Verlag, 1999.

[17] A. D. Gordon and A. Jeffrey. Authenticity by Typing for Security Protocols. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW 2001)*, pages 145–159. IEEE Computer Society, 2001.

[18] C. A. Meadows. The NRL protocol analyzer: an overview. In *Proc. of the 2nd International Conference on the Practical Application of PROLOG*, 1994.

[19] D. Miller. Higher-order quantification and proof search. In *Proceedings of the AMAST confrerence*, LNCS. Springer, 2002.

[20] R. Milner. *Communication and Concurrency.* International Series in Computer Science. Prentice Hall, 1989.

[21] R. Milner. *Communicating and Mobile Systems: the π-Calculus.* Cambridge University Press, 2000.

[22] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, I and II. *Information and Computation*, 100(1):1–40, 1992.

[23] S. Schneider. Security properties and CSP. In *Proc. of the IEEE Symposium on Research in Security and Privacy*, pages 174–187, 1996.

[24] S. Schneider. Verifying Authentication Protocols in CSP. *IEEE Transaction on Sofware Engineering*, 24(8):743–758, 1998.

[25] J. Thayer, J. Herzog, and J. Guttman. Honest ideals on strand spaces. In *Proc. of the 11th IEEE Computer Security Foundations Workshop (CSFW '98)*, pages 66–78, Washington - Brussels - Tokyo, 1998. IEEE.

[26] J. Thayer, J. Herzog, and J. Guttman. Strand spaces: Why is a security protocol correct? In *Proc. of the 19th IEEE Computer Society Symposium on Research in Security and Privacy*, 1998.