Relating Process Algebras and Multiset Rewriting for Immediate Decryption Protocols

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 mappings 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 CCS and the π-calculus. Although defining a simple and general bijection between MSR and PA appears difficult, we show that the sublanguages needed to specify a large class of cryptographic protocols (immediate decryption protocols) admits an effective translation that is not only bijective and trace-preserving, but also induces a weak form of bisimulation across the two languages. In particular, the correspondence sketched in this abstract permits transferring several important trace-based properties such as secrecy and many forms of authentication.

* 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.,* [6, 19, 15, 25, 27, 3, 9, 7, 12, 14, 16, 18, 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 [17], and to study the relationships between different models for representing security protocols [8].

In this paper, we relate uses of process algebras (PA) and multiset-rewriting (MSR) for the description and the analysis of a large class of security protocols by defining *encodings* from one formalism to the other. A general comparison between PA and MSR yields results weaker than the specialized application to security protocols analyzed here [4]. Differently from [5], we restrict our attention to a subclass of protocols that never receive a key needed to decrypt an earlier message. We call them "immediate decryption protocols". Although this class encompasses most protocols studied in the literature, and therefore is interesting on its own, a more general treatment is given in [4, 5].

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 [22] and CCS [21]. We expect that it will be possible to adapt our results to other (value passing) process algebras used in security protocol analysis, *e.g.,* the *spi*-calculus [2] or even CSP [24]. Indeed, 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 [7]. It is also playing a practical role as the CIL intermediate language [12] of the CASPL security protocol analysis system [11], 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 [26], a popular specification language for crypto-protocols, have been analyzed in [8].

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. In particular, the simple correspondence achieved in this paper is sufficient to transfer several useful trace-based properties such as secrecy and many forms of authentication. Second, by bridging PA and MSR, we implicitly define a correspondence between PA and other languages, in particular strand spaces [26] (in a setting with an

interleaving semantics), a worthy investigation as remarked in [10]. Interesting work about linear logic and multiset rewriting appears in [20].

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 vice versa (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:

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

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:

$$\begin{array}{ll} \textit{Single rule application} & \tilde{r} : \tilde{a} \longrightarrow \tilde{b} \\ \textit{Iterated rule application} & \tilde{r} : \tilde{a} \longrightarrow^* \tilde{b} \end{array}$$

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:

$$\begin{array}{lrcl} \textit{Parallel processes} & Q & ::= & 0 \mid Q \parallel P \mid Q \parallel !P \\ \textit{Sequential processes} & P & ::= & 0 \mid \overline{a}(\boldsymbol{t}).P \mid a(\boldsymbol{t}).P \mid \nu x.P \end{array}$$

Parallel processes are defined as a parallel composition of, possibly replicated, sequential processes. These, in turn, are a sequence of communication actions (input or output) and constant generation. An output process $\overline{a}(\boldsymbol{t}).P$ is ready to send a tuple $\boldsymbol{t} = (t_1, \ldots, t_k)$, built over a signature Σ, along the polyadic channel named a. An input process $a(\boldsymbol{t}).P$ is ready to receive a tuple of messages and match them against the patterns $\boldsymbol{t}$, possibly binding previously unbound variables in it. Finally, the creation of a new object in P (as in the π-calculus [23]) is written as $\nu x.P$. The binders of our language are $\nu x.$ and $a(\boldsymbol{t})$, which bind x and any first occurrence of a variable in $\boldsymbol{t}$, respectively. This induces the usual definition of free and bound variables in a term or process. We write $var(t)$ for the free symbols occurring in a process Q; no channel name is ever free. This language is sufficiently expressive to conveniently formalize immediate decryption protocols. The general case, which makes use of delayed decryption, requires an explicit pattern matching operator. A preliminary solution to this more general problem is presented in [5], while a proper treatment appears in [4].

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:

$$\frac{\boldsymbol{t} = \boldsymbol{t'}[\theta]}{(Q \parallel \overline{a}(\boldsymbol{t}).P \parallel a(\boldsymbol{t'}).P') \Rightarrow (Q \parallel P \parallel P'[\theta])} \qquad \frac{k \notin var(Q, P)}{(Q \parallel \nu x.P) \Rightarrow (Q \parallel P[k/x])}$$

$$\frac{Q \equiv Q'' \quad Q'' \Rightarrow Q'}{Q \Rightarrow Q'} \qquad \frac{}{Q \Rightarrow^* Q} \qquad \frac{Q \Rightarrow Q'' \quad Q'' \Rightarrow^* Q'}{Q \Rightarrow^* Q'}$$

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 patterns $\boldsymbol{t'}$ react by applying the instantiating substitution θ to P'. The second rule defines the semantics of νx as instantiation with an eigenvariable. The next rule allows interactions to happen modulo structural equivalence, $\equiv$, that in our case contains the usual monoidal equalities of parallel processes with respect to $\parallel$ and 0, and the unfolding of replication (*i.e.,* , $!P = !P \parallel P$). Finally, the last two inferences define $\Rightarrow^*$ as the reflexive and transitive closure of $\Rightarrow$.

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, 17]). 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 that includes at least concatenation ($\langle \_, \_ \rangle$) and encryption ($\{\_\}_\_$). 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.

3.1 Protocols as Multiset Rewriting

MSR_P relies on the following predicate symbols [8]:

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, the family of *role state predicates* $\{A_{\rho_i}(\boldsymbol{t}) : i = 0 \dots l_\rho\}$, 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* $\quad r_{\rho_0} : \tilde{\pi}(\boldsymbol{x}) \rightarrow \exists \boldsymbol{n}.A_{\rho_0}(\boldsymbol{n}, \boldsymbol{x}), \tilde{\pi}(\boldsymbol{x})$
- zero or more $(i = 1 \dots l_\rho)$ *message exchange rules*:

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

Notice that, differently from [5] where delayed decryption protocols are handled, the role state predicates are restricted to having only variables as their arguments. The notation $t(\boldsymbol{z})$ is meant to indicate that the variables appearing in term t are drawn from $\boldsymbol{z}$. We can safely elide $A_{\rho_{l_\rho}}(\boldsymbol{x})$ from the very last rule.

It should be observed that the form of these rules does not allow representing protocols that receive an encrypted message component and only at a later stage

are given the key to decrypt it. This case requires a more complicated treatment, which is presented in [5] and fully analyzed in [4]. The simplified study analyzed here should not be dismissed, however, since the majority of the protocols studied in the literature do not manifest the above behavior.

Rules in $\tilde{r}_I$ are the standard rules describing the intruder in the style of Dolev-Yao [13], whose capabilities consist in intercepting, analyzing, synthesizing and constructing messages, with the ability to access permanent data. Formally:

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

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 $(\tilde{r} : \tilde{s}_0)$. An example specification is given in Appendix A.1.

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}); (c) 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, defined as the parallel composition of five components: $P_{!net} \parallel \prod_\rho P_{!\rho} \parallel Q_{!I} \parallel Q_{!\pi} \parallel Q_{I_0}$ where $\prod \mathcal{P}$ denotes the parallel composition of all the processes in $\mathcal{P}$. More 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.

$P_{!\rho}$**.** 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[1]: $P_\rho = !\tilde{\pi}(\boldsymbol{x}).\nu\boldsymbol{n}.P'_\rho$ where P'_ρ is a sequential process that does input and output only on the network channels. Formally, $P'_\rho ::= 0 \mid N_o(\boldsymbol{t}).P'_\rho \mid \overline{N_i}(\boldsymbol{t}).P'_\rho$. An example specification is given in Appendix A.2.

[1] 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 $\nu n_1.\ldots.\nu n_h.P$.

Again, this definition does not allows protocols that need remembering encrypted messages until they are given the key. This form of specification, which requires to enrich PA with a pattern matching construct, is analyzed in [4, 5].

$Q_{!I} = !P_{I_1} \parallel \ldots \parallel !P_{I_{10}}$. 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:

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

Notice that, in our simple calculus, the decomposition of pairs shall be treated as two projection rules (P'_{I_5} and P''_{I_5}). See [4] for a detailed discussion.

$Q_{!\pi} = \prod !\overline{\pi}(t).0$ 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.,* $\mathtt{Kp}$). It is assumed that no other process performs an output on π.

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

In PA_P, a *state* is a process of the form $Q_! \parallel Q_{net} \parallel \prod_\rho P_\rho \parallel Q_I$ where $Q_! = (P_{!net} \parallel \prod_\rho !P_\rho \parallel Q_{!I} \parallel Q_{!\pi})$ while Q_{net}, P_ρ and Q_I are ground, unreplicated (*i.e.,* without a bang as a prefix) instances of sequential suffixes of processes $P_{!net}$ (more precisely $\overline{N_o}(t).0$), $\prod_\rho P_{!\rho}$, and $P_{!I}$ (more precisely $\overline{I}(t).0$ or $\overline{I}(t).\overline{I}(t).0$).

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 the presence of multiple instances of a same role (they are addressed through replicated processes). 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.

4.1 From MSR_P to PA_P

Given an MSR_P configuration $(\tilde{r} : \tilde{s})$, with $\tilde{r} = (\cup_\rho(\tilde{r}_\rho), \tilde{r}_I)$ and $\tilde{s} = (\tilde{A}, \tilde{N}, \tilde{I}, \tilde{\pi})$, we return a PA_P state $Q_! \parallel Q_{net} \parallel \prod_{\tilde{A}} P_{\tilde{A}} \parallel Q_I$ (with $Q_! = (P_{!net} \parallel \prod_\rho P_{!\rho} \parallel Q_{!I} \parallel Q_{!\pi})$). 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) $Q_{!\pi}$ results from the transformation of $\tilde{\pi}$, and (d) $\prod_{\tilde{A}} P_{\tilde{A}}$, Q_{net}, and Q_I result from transformation of $\tilde{A}$, $\tilde{N}$ and $\tilde{I}$, respectively.

Processes $P_{!\rho}$, 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 $0 < i \leq l_\rho - 1$:

$$\lceil r_{\rho_{i+1}} \rceil = \begin{cases} \overline{N_i}(t(\boldsymbol{x})).\lceil r_{\rho_{i+2}} \rceil & , \text{ if } r_{\rho_{i+1}} = A_{\rho_i}(\boldsymbol{x}) \rightarrow A_{\rho_{i+1}}(\boldsymbol{x}), N(t(\boldsymbol{x})) \\ N_o(t(\boldsymbol{x}, \boldsymbol{y}))\lceil r_{\rho_{i+2}} \rceil, & \text{ if } r_{\rho_{i+1}} = A_{\rho_i}(\boldsymbol{x}), N(t(\boldsymbol{x}, \boldsymbol{y})) \rightarrow A_{\rho_{i+1}}(\boldsymbol{x}, \boldsymbol{y}) \end{cases}$$

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: $P_\rho \stackrel{def}{=} \lceil r_{\rho_0} \rceil$

The intruder is handled by simply mapping $\tilde{r}_I$ to $Q_{!I}$. More precisely, we define the transformation function $\lceil\_\rceil_I$ that relates the intruder rewriting rule r_{I_j} with the sequential agents P_{I_j} defined in Section 3.2 (r_{I_5} is mapped to the pair P'_{I_5} and P''_{I_5}). 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 the state $\tilde{s} = (\tilde{A}, \tilde{N}, \tilde{I}, \tilde{\pi})$ is treated. For each $A_{\rho_i}(\boldsymbol{t}) \in \tilde{A}$, we define $P_{A_{\rho_i}(\boldsymbol{t})} = \lceil r_{\rho_{i+1}} \rceil[\boldsymbol{x}/\boldsymbol{t}]$, where $\boldsymbol{x}$ are the variables appearing as argument of A_i in $r_{\rho_{i+1}}$.

The multiset $\tilde{N}$ guides the definition of Q_{net}, that is $Q_{net} \stackrel{def}{=} \prod_{N(t) \in \tilde{N}} \overline{N}(t).0$. Similarly, $Q_I \stackrel{def}{=} \prod_{I(t) \in \tilde{I}} \overline{I}(t).0$. On the other hand, $Q_{!\pi} \stackrel{def}{=} \prod_{\pi(t) \in \tilde{\pi}} !\overline{\pi}(t).0$.

Writing, with a little abuse of notation, $\lceil\_\rceil$ for the generic function that, given a multiset rewriting MSR_P configuration returns a PA_P state, the encoding can be summarized as:

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

4.2 From PA_P to MSR_P

Given a PA_P state $Q_! \parallel Q_{net} \parallel \prod_\rho P_\rho \parallel Q_I$ with $Q_! = (P_{!net} \parallel \prod_\rho P_{!\rho} \parallel Q_{!I} \parallel Q_{!\pi})$, we show how to construct a configuration in MSR_P. The basic translation involves the transformation function $\lfloor \_ \rfloor^{\#}_{(i;\epsilon)}$ for the $P_{!\rho}$'s (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:

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

Let $P_{!\rho}$ be the role specification of which an object P_ρ in $\prod_\rho P_\rho$ is an instantiated suffix and $\theta = [\boldsymbol{x}/\boldsymbol{t}]$ the witnessing substitution. If P_ρ starts with either a persistent input $\pi(\boldsymbol{x})$ or the ν operator, we set $\lfloor P_\rho \rfloor = \cdot$. Otherwise, let i be the index at which P_ρ occurs in $P_{!\rho}$ as for the above definition. Then $\lfloor P_\rho \rfloor = A_{\rho i}(\boldsymbol{t})$.

The intruder process $Q_{!I}$ is roughly handled by the inverse of the transformation $\lceil \_ \rceil_I$, which we call $\lfloor \_ \rfloor_I$ (see [4] for a more detailed and rigorous treatment). Each object $\overline{I}(t).0$ (resp., the $\overline{I}(t).\overline{I}(t).0$) in Q_I is rendered as the state element $I(t)$ (resp., pair of elements $I(t), I(t)$).

$P_{!net}$ disappears. Instead, each occurrence of a process $\overline{N_o}(t).0$ in P_{net} is mapped to a state element $N(t)$. Similarly, each process $!\pi(\boldsymbol{x})$ in $P_{!\pi}$ is translated into the state object $\pi(\boldsymbol{x})$.

It is easy to prove that $\lceil \_ \rceil$ and $\lfloor \_ \rfloor$ are inverse of each other:

Lemma 1.

1. *For any* MSR_P *configuration* C*, we have that* $\lfloor \lceil C \rceil \rfloor = C$.
2. *For any* PA_P *state* Q*, we have that* $\lceil \lfloor Q \rfloor \rceil = Q$ *modulo the extension of* $\equiv$ *with the equation* $\overline{a}(\boldsymbol{t}).\overline{a}(\boldsymbol{t}).0 = \overline{a}(\boldsymbol{t}).0 \parallel \overline{a}(\boldsymbol{t}).0$.

A detailed proof of this result, including the treatment of details that have been omitted here for space reasons, can be found in [4].

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. This will allow us to prove that the translations presented in this paper are reachability-preserving in a very strong sense. We invite the reader to consult [4] for a more comprehensive and detailed discussion of this matter.

We first formalize the notion of transition step and observation in each formalism.

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

$$\frac{}{0 \xrightarrow{\cdot}} \qquad \frac{Q \xrightarrow{\alpha} \qquad P \xrightarrow{\alpha'}}{(Q \parallel P) \xrightarrow{\alpha,\alpha'}} \qquad \frac{}{\nu n.P \xrightarrow{\cdot}} \qquad \frac{}{a(\boldsymbol{t}).P \xrightarrow{a(\boldsymbol{t})}} \qquad \frac{}{\overline{a}(\boldsymbol{t}).P \xrightarrow{\overline{a}(\boldsymbol{t})}}$$

We write $\alpha \in P$ if $\alpha \in \{\alpha : P \xrightarrow{\alpha'}\}$.

Let c be a channel name (a) or the complement of a channel name ($\overline{a}$), we define the observations *of process Q along c as the multiset $Obs_c(Q) = \{\boldsymbol{t} : c(\boldsymbol{t}) \in Q\}$.*

Definition 2. *Given a multiset of ground atoms $\tilde{s}$ and a predicate name a, we define the* projection *of $\tilde{s}$ along a as the multiset $Prj_a(\tilde{s}) = \{\boldsymbol{t} : a(\boldsymbol{t}) \in \tilde{s}\}$.*

If $C = (\tilde{r}; \tilde{s})$ is a configuration, we set $Prj_a(\tilde{C}) = Prj_a(\tilde{s})$.

Using Definitions 1 and 2, we make precise what we intend for an MSR_P configuration and a PA_P state to be corresponding.

Definition 3. *Given an MSR_P configuration C and a PA_P state Q. We say that C and Q are* corresponding*, written as $C \bowtie Q$, if and only if the two conditions hold:*

1. $Prj_N(C) = Obs_{\overline{N_o}}(Q)$
2. $Prj_I(C) = Obs_{\overline{I}}(Q)$

Informally $C \bowtie Q$ means that the messages that are lying on the net and the intruder knowledge are the same in configuration C and state Q.

On the basis of these concepts, we can now define a correspondence between MSR_P configurations and PA_P states such that, if in MSR_P is possible to perform an action (by applying a rule) that will lead to a new configuration, then in PA_P is possible to follow some transitions that will lead in a corresponding state, and vice versa.

Definition 4. *Let $\mathcal{C}$ and $\mathcal{Q}$ be the set of all MSR_P configurations and PA_P states, respectively. A binary relation $\sim \subseteq \mathcal{C} \times \mathcal{Q}$ is a* correspondence *if $(\tilde{r} : \tilde{s}) \sim Q$ implies that:*

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

The following theorems, whose proof can be found in [4], affirm that security protocol specifications written in MSR_P and PA_P, and related via the encodings here presented, are corresponding. The treatment of the intruder requires some care, that, for space reasons, we have partially hidden from the reader by presenting a slightly simplified translation of $Q_{!I}$ into $\tilde{r}_I$. Again, all details can be found in [4].

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

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

This means that any step in MSR_P can be faithfully simulated by zero or more steps in PA_P through the mediation of the encoding $\lceil \_ \rceil$, and vice-versa, the reverse translation $\lfloor \_ \rfloor$ will map steps in PA_P into corresponding steps in MSR_P.

6 Conclusions

This paper shows how multiset rewriting theories (MSR_P) and process algebras (PA_P) used to describe the large class of immediate decryption protocols may be related. Indeed we show how to define transformations between MSR_P to PA_P specifications of such 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 network and on what messages the intruder knows. A direct consequence of this results is that any confidentiality property established in one framework can automatically be ported to the other. Moreover, since several forms of authentication among protocol participants may be formulated in terms of properties of what is sent to the net or what is captured by the intruder, authentication results can also also immediately transfered through our encodings.

Acknowledgments

We would like to thank the selection committee and attendees of the WITS'03 workshop where a preliminary version of this paper was presented [5]. Their feedback gave us precious material for thought and led to the present revision, which focuses on immediate decryption protocols, while the complications of the general case are addressed in the technical report [4].

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 security protocol analysis. Technical report, ISTI-CNR, 2003. Available at `http://matrix.iei.pi.cnr.it/~lenzini/pub/msr.ps`.

[5] Stefano Bistarelli, Iliano Cervesato, Gabriele Lenzini, and Fabio Martinelli. Relating Process Algebras and Multiset Rewriting for Security Protocol Analysis. In R. Gorrieri, editor, *Third Workshop on Issues in the Theory of Security — WITS'03*, Warsaw, Poland, 2003.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

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

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

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

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

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

[24] S. Schneider. Security properties and CSP. In *Proc. of the IEEE Symposium on Research in Security and Privacy*, pages 174–187, 1996.
[25] S. Schneider. Verifying Authentication Protocols in CSP. *IEEE Transaction on Sofware Engineering*, 24(8):743–758, 1998.
[26] 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.
[27] 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.

A The Needham-Schroeder Public-Key Protocol

To help understand the detail of the specification and of the mappings, we will analyze an example, consisting of the following classical protocol:

$$\begin{array}{l} A \longrightarrow B : \{A, N_A\}_{K_B} \\ B \longrightarrow A : \{B, N_A, N_B\}_{K_A} \\ A \longrightarrow B : \{N_B\}_{K_B} \end{array}$$

A.1 NSPK in MSR$_P$

The MSR$_P$ specification of this protocol will consist of the rule-set $\mathcal{R}_{\text{NSPK}} = (\mathcal{R}_A, \mathcal{R}_B)$. $\mathcal{R}_A$ and $\mathcal{R}_B$ implement the roles of the initiator (A) and the responder (B) respectively. They are given as follows:

$$\mathcal{R}_A \begin{cases} & \tilde{\pi}(\boldsymbol{A}) & \rightarrow \exists N_A.\tilde{\pi}(\boldsymbol{A}), & A_0(\boldsymbol{A}, N_A) \\ A_0(\boldsymbol{A}, N_A) & & \rightarrow N(\{A, N_A\}_{K_B}), & A_1(\boldsymbol{A}, N_A) \\ A_1(\boldsymbol{A}, N_A), & N(\{B, N_A, N_B\}_{K_A}) & \rightarrow & A_2(\boldsymbol{A}, N_A, N_B) \\ A_2(\boldsymbol{A}, N_A, N_B) & & \rightarrow N(\{N_B\}_{K_B}), & A_3(\boldsymbol{A}, N_A, N_B) \end{cases}$$

$$\mathcal{R}_B \begin{cases} & \tilde{\pi}(\boldsymbol{B}) & \rightarrow \exists N_B.\tilde{\pi}(\boldsymbol{B}), & B_0(\boldsymbol{B}, N_B) \\ B_0(\boldsymbol{B}, N_B), & N(\{A, N_A\}_{K_B}) & \rightarrow & B_1(\boldsymbol{B}, N_B, N_A) \\ B_1(\boldsymbol{B}, N_B N_A) & & \rightarrow N(\{B, N_A, N_B\}_{K_A}), & B_2(\boldsymbol{B}, N_B, N_A) \\ B_2(\boldsymbol{B}, N_B, N_A), & N(\{N_B\}_{K_B}) & \rightarrow & B_3(\boldsymbol{B}, N_B, N_A) \end{cases}$$

where:
$$\begin{array}{l} \boldsymbol{A} = (A, B, K_A, K'_A, K_B) \\ \boldsymbol{B} = (B, A, K_B, K'_B, K_A) \end{array}$$

$$\tilde{\pi}(X, Y, K_X, K'_X, K_Y) = \mathtt{Pr}(X), \mathtt{PrK}(X, K'_X), \mathtt{PbK}(Y, K_Y), \mathtt{Kp}(K_X, K'_X)$$

In addition, we assume the state portion of a configuration consists of:

$$\underbrace{\tilde{\pi}(a, b, k_a, k'_a, k_b), \tilde{\pi}(b, e, k_b, k'_b, k_e), \tilde{\pi}(e, a, k_e, k'_e, k_a)}_{\tilde{\pi}}, \underbrace{I(e), I(k_e), I(k'_e)}_{\tilde{I}} \underbrace{}_{\tilde{N}} \underbrace{}_{\tilde{A}}$$

where a, b, e, $k_{\_}$ and $k'_{\_}$ are constants (e stands for the intruder).

A.2 NSPK in PA_P

$$NSPK = P_{!net} \parallel Q_{!I} \parallel P_{!A} \parallel P_{!B} \parallel Q_{!\pi} \parallel Q_{I_0}$$

$P_{!net}$ and $Q_{!I}$ have already been defined. The other processes are as follows:

- $P_{!A} = !\tilde{\pi}(\boldsymbol{A}).\nu N_A.\overline{N_i}(\{A, N_A\}_{K_B}).N_o(\{B, N_A, N_B\}_{K_A}).\overline{N_i}(\{N_B\}_{K_B}).0$.
- $P_{!B} = !\tilde{\pi}(\boldsymbol{B}).\nu N_B.N_o(\{A, N_A\}_K).\overline{N_i}(\{B, N_A, N_B\}_{K_A}).N_o(\{N_B\}_{K_B}).0$
  where, as for MSR_P, $\boldsymbol{A} = (A, B, K_A, K'_A, K_B)$ and $\boldsymbol{B} = (B, A, K_B, K'_B, K_A)$, and $\tilde{\pi}(X, Y, K_X, K'_X, K_Y)$ inputs each of the object in $\tilde{\pi}(X, Y, K_X, K'_X, K_Y)$ (see A.1). More precisely, it is defined as

  $$\mathtt{Pr}(X).\mathtt{PrK}(X, K'_X).\mathtt{PbK}(Y, K_Y).\mathtt{Kp}(K_X, K'_X)\,.$$

- $Q_{!\pi} = Q_{\tilde{\pi}(a,b,k_a,k'_a,k_b)} \parallel Q_{\tilde{\pi}(b,e,k_b,k'_b,k_e)} \parallel Q_{\tilde{\pi}(e,a,k_e,k'_e,k_a)}$
  where $Q_{\tilde{\pi}(X,Y,K_X,K'_X,K_Y)}$ is the parallel composition of simple replicated processes that output each object in $\tilde{\pi}(X, Y, K_X, K'_X, K_Y)$ on channel π:

  $$!\overline{\mathtt{Pr}}(X).0 \parallel !\overline{\mathtt{PrK}}(X, K'_X).0 \parallel !\overline{\mathtt{PbK}}(Y, K_Y).0 \parallel !\overline{\mathtt{Kp}}(K_X, K'_X).0\,.$$

- $Q_{I_0} = \overline{I}(e).0 \parallel \overline{I}(k_e).0 \parallel \overline{I}(k'_e).0$.