A Comparison between Strand Spaces and Multiset Rewriting for Security Protocol Analysis* **

I. Cervesato[1], N. Durgin[2], P. Lincoln[3], J. Mitchell[2], and A. Scedrov[4]

[1] Advanced Engineering and Sciences Division, ITT Industries, Inc.
2560 Huntington Avenue, Alexandria, VA 22303-1410 — USA
iliano@itd.nrl.navy.mil

[2] Computer Science Department, Stanford University
Stanford, CA 94305-9045 — USA
{nad, jcm}@cs.stanford.edu

[3] Computer Science Laboratory, SRI International
333 Ravenswood Avenue, Menlo Park, CA 94025-3493 — USA
lincoln@csl.sri.com

[4] Mathematics Department, University of Pennsylvania
209 South 33rd Street, Philadelphia, PA 19104-6395 — USA
scedrov@cis.upenn.edu

Abstract. Formal analysis of security protocols is largely based on a set of assumptions commonly referred to as the Dolev-Yao model. Two formalisms that state the basic assumptions of this model are related here: strand spaces [FHG98] and multiset rewriting with existential quantification [CDL+99,DLMS99]. Strand spaces provide a simple and economical approach to state-based analysis of completed protocol runs by emphasizing causal interactions among protocol participants. The multiset rewriting formalism provides a very precise way of specifying finite-length protocols, with a bounded initialization phase but allowing unboundedly many instances of each protocol role, such as client, server, initiator, or responder. Although it is fairly intuitive that these two languages should be equivalent in some way, a number of modifications to each system are required to obtain a meaningful equivalence. We extend the strand formalism with a way of incrementally growing bundles in order to emulate an execution of a protocol with parametric strands. We omit the initialization part of the multiset rewriting setting, which formalizes the choice of initial data, such as shared public or private keys, and which has no counterpart in the strand space setting. The correspondence between the modified formalisms directly relates the intruder theory from the multiset rewriting formalism to the penetrator strands. The relationship we illustrate here between multiset rewriting specifications and strand spaces thus suggests refinements to both frameworks, and deepens our understanding of the Dolev-Yao model.

* Partial support for various authors by OSD/ONR CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing" through ONR Grant N00014-01-1-0795, by NRL under contract N00173-00-C-2086, by DoD MURI "Semantic Consistency in Information Exchange" as ONR grant N00014-97-1-0505 and by NSF grants CCR-9509931, CCR-9629754, CCR-9800785, CCR-0098096, and INT98-15731.

** This paper is a revised version of [CDL+00].

1 Introduction

Security protocols are widely used to protect access to computer systems and to protect transactions over the Internet. Such protocols are difficult to design and analyze for several reasons. Some of the difficulties come from subtleties of cryptographic primitives. Further difficulties arise because security protocols are required to work properly when multiple instances of the protocol are carried out in parallel, where a malicious intruder may combine data from separate sessions in order to confuse honest participants. A variety of methods have been developed for analyzing and reasoning about security protocols. Most current formal approaches use the so-called Dolev-Yao model of adversary capabilities, which appears to be drawn from positions taken in [NS78] and from a simplified model presented in [DY83]. The two basic assumptions of the Dolev-Yao model, perfect (black-box) cryptography and a nondeterministic adversary, provide an idealized setting in which protocol analysis becomes relatively tractable.

One recent setting for stating the basic assumptions of the Dolev-Yao model is given by strand spaces [FHG98,FHG99,Man99]. Strand spaces provide a way of presenting information about causal interactions among protocol participants. Roughly, a strand is a linearly ordered sequence of events that represents the actions of a protocol participant. A strand space is a collection of strands, equipped with a graph structure generated by causal interaction. Strand spaces provide a simple and succinct framework for state-based analysis of completed protocol runs. State space reduction techniques based on the strand space framework are utilized in an efficient automated checker, Athena [Son99].

Protocol transitions may also be naturally expressed as a form of rewriting. This observation may be sharpened to a rigorous, formal definition of the Dolev-Yao model by means of multiset rewriting with existential quantification [CDL$^+$99,DLMS99]. In this framework protocol execution may be carried out symbolically. Existential quantification, as commonly used in formal logic, provides a natural way of choosing new values, such as new keys or nonces. Multiset rewriting provides a very precise way of specifying security protocols and has been incorporated into a high-level specification language for authentication protocols, CAPSL [DM99]. As presented in [CDL$^+$99,DLMS99], a protocol theory consists of three parts: a bounded phase describing protocol initialization that distributes keys or establishes other shared information, a role generation theory that designates possibly multiple roles that each principal may play in a protocol (such as initiator, responder, or server), and a disjoint union of bounded subtheories that each characterize a possible role. The multiset rewriting formalism allows us to formulate one standard intruder theory that describes any adversary for any protocol.

One would expect that strand spaces and multiset rewriting should be equivalent in some way. However, a meaningful equivalence may be obtained only after a number of modifications are made in each setting. To this end, we extend the strand space setting by introducing several dynamic concepts that describe the evolution of parametric strands as an execution of a protocol unfolds. In particular, we present a formalized notion of parametric strands and we describe a way of incrementally growing strand space bundles in order to emulate an execution of a protocol with parametric strands. In addition to contributing to the understanding of the strand space setting, these extensions make possible the comparison with multiset rewriting specifications. In order to

obtain a precise equivalence, we also must drop the initialization part of the multiset rewriting formalism, which specifies the choice of initial conditions. In many protocols, the initial conditions specify generation of fresh shared, public, or private keys. The initialization phase generating fresh initial data has no counterpart in the strand space setting. We also anticipate the validation of variable instantiations to the very beginning of the execution of a role. After these modifications, there is a straightforward and direct correspondence between strand spaces and multiset rewriting theories. Moreover, the correspondence directly relates the intruder theories from the multiset rewriting formalism to penetrator strands. We believe that the investigation of the exact nature of the relationship between the two formalisms deepens our understanding of the Dolev-Yao model and can suggest extensions and refinements to these and other specification languages based on strand spaces.

Shortly after publishing the first version of this paper, the authors noticed an error in Lemma 4.1 in [CDL+00]. The corrected version of this lemma is Corollary 1 in the present paper, which rectifies the error and propagates the resulting changes. It also reorganizes the material into a more clear and straightforward presentation.

This paper is organized as follows: the multiset rewriting formalism is discussed in Section 2. In section 3, we discuss strand spaces and present our extensions. The translation from multiset rewriting to strand spaces is presented in Section 4. The translation from strand spaces to multiset rewriting is presented in Section 5.

2 Multiset Rewriting Theories

In Section 2.1 we recall a few multiset rewriting concepts, and, in Section 2.2, we apply them to the specification of cryptoprotocols. We present the multiset rewriting rules implementing the Dolev-Yao intruder in Section 2.3.

2.1 Multiset Rewriting

A *multiset* $\boldsymbol{M}$ is an unordered collection of objects or *elements*, possibly with repetitions. The *empty multiset* does not contain any object and will be written "$\cdot$". We accumulate the elements of two multisets $\boldsymbol{M}$ and $\boldsymbol{N}$ by taking their *multiset union*, denoted "$\boldsymbol{M}, \boldsymbol{N}$". The elements we will consider here will be first-order atomic formulas $A(\boldsymbol{t})$ over some signature.

We will make use of the standard definitions pertaining to the variables of first-order logic. In particular, we write $\mathrm{Var}(A_0, \ldots, A_n)$ for the set of variables occurring in the multiset of atomic formulas $A_0, \ldots, A_n$. We say that a (multiset of) formula(s) is ground if no variable appear in it. Finally, substitutions (generally written θ) are as usual mapping from variables to generic terms. We write $A[\theta]$ for the application of a substitution θ to a formula A, and use a similar notation for multisets of formulas.

In its simplest form, a *multiset rewrite rule* r is a pair of multisets $\boldsymbol{F}$ and $\boldsymbol{G}$, respectively called the *antecedent* and *consequent* of r. We will consider a slightly more elaborate notion in which $\boldsymbol{F}$ and $\boldsymbol{G}$ are multisets of first-order atomic formulas with variables among $\boldsymbol{x}$. We emphasize this aspect by writing them as $\boldsymbol{F}(\boldsymbol{x})$ and $\boldsymbol{G}(\boldsymbol{x})$.

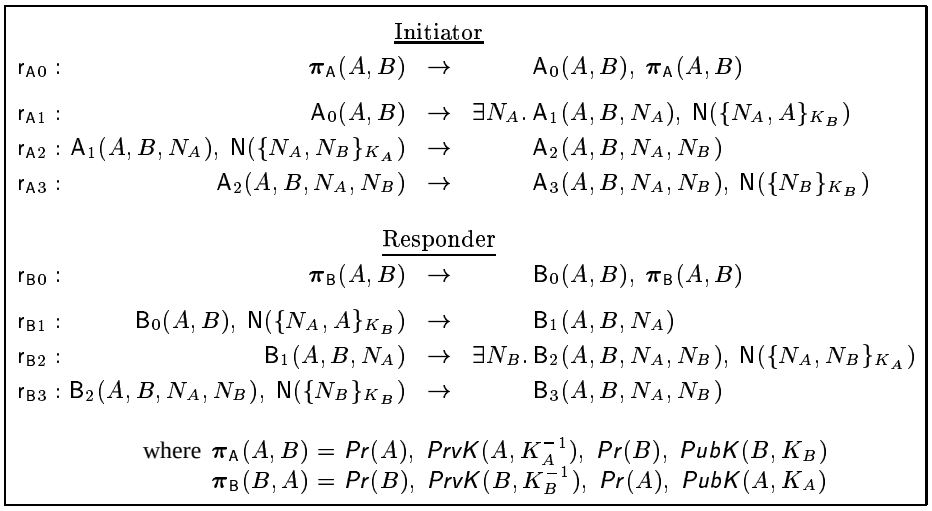


Fig. 1. Multiset Rewriting Specification of the Needham-Schroeder Protocol

Furthermore, we shall be able to mark variables in the consequent so that they are instantiated to "*fresh*" constants, that have not previously been encountered, even if the rule is used repeatedly. A rule assumes then the form

$$r:\ \boldsymbol{F}(\boldsymbol{x}) \rightarrow \exists \boldsymbol{n}.\, \boldsymbol{G}(\boldsymbol{x},\boldsymbol{n})$$

where r is a label and $\exists \boldsymbol{n}$ indicates that the variables $\boldsymbol{n}$ are to be instantiated with constants that ought to be fresh. A *multiset rewriting system* $\mathcal{R}$ is a set of rewrite rules.

Rewrite rules allow transforming a multiset into another multiset by making localized changes to the elements that appear in it. Given a multiset of ground facts $\boldsymbol{M}$, a rule $r:\ \boldsymbol{F}(\boldsymbol{x}) \rightarrow \exists \boldsymbol{n}.\, \boldsymbol{G}(\boldsymbol{x},\boldsymbol{n})$ is *applicable* if $\boldsymbol{M} = \boldsymbol{F}(\boldsymbol{t}), \boldsymbol{M}'$, for terms $\boldsymbol{t}$. Then, *applying* r to $\boldsymbol{M}$ yields the multiset $\boldsymbol{N} = \boldsymbol{G}(\boldsymbol{t},\boldsymbol{c}), \boldsymbol{M}'$ where the constants $\boldsymbol{c}$ are fresh (in particular, they do not appear in $\boldsymbol{M}$), $\boldsymbol{x}$ and $\boldsymbol{n}$ have been instantiated with $\boldsymbol{t}$ and $\boldsymbol{c}$ respectively, and the facts $\boldsymbol{F}(\boldsymbol{t})$ in $\boldsymbol{M}$ have been replaced with $\boldsymbol{G}(\boldsymbol{t},\boldsymbol{c})$ to produce $\boldsymbol{N}$. Here, $\theta = [\boldsymbol{t}/\boldsymbol{x}, \boldsymbol{c}/\boldsymbol{n}]$ is the *instantiating substitution* of rule r with respect to $\boldsymbol{M}$. We denote the application of a single rule and of zero or more rewrite rules by means of the *one-step* and *multistep transition* judgments:

$$\boldsymbol{M} \xrightarrow{r}_{\mathcal{R}} \boldsymbol{N} \qquad\qquad \boldsymbol{M} \xrightarrow{\boldsymbol{r}}_{\mathcal{R}}^{*} \boldsymbol{N}$$

respectively. The labels r and $\boldsymbol{r}$ identify which rule(s) have been applied together with its (their) instantiating substitution(s). Thus, $\boldsymbol{r}$ acts as a complete trace of the execution.

2.2 Regular Protocol Theories

We model protocols by means of specifically tailored multiset rewriting systems that we call *regular protocol theories*. We present here a simplified version of the model introduced in [CDL+99,DLMS99]. We rely upon the following atomic formulas:

Persistent information: Data such as the identity of principals and their keys often constitute the stage on which the execution of a protocol takes place, and does not change as it unfolds. We will represent and access this *persistent information* through a fixed set of *persistent predicates* that we will indicate using a slanted font (*e.g.* KeyP, as opposed to N).

In [CDL$^+$99,DLMS99], we described the choice of the persistent data by means of a set of multiset rewrite rules of a specific form, that we called the *initialization theory*. We showed that the application of these rules can be confined to an initialization phase that precedes the execution of any other rule. Let $\boldsymbol{\Pi}$ be the resulting set of ground facts (constraints on the initialization theory prevent $\boldsymbol{\Pi}$ from containing duplicates [CDL$^+$99,DLMS99]). Strand constructions assume instead that the persistent information is given up-front as a set. We reconcile the two approaches by dropping the explicit initialization phase of [CDL$^+$99,DLMS99] and assuming $\boldsymbol{\Pi}$ given. We will allow individual rules to query $\boldsymbol{\Pi}$ (but not to modify it).

Network messages: Network messages are modeled by the predicate $\mathsf{N}(m)$, where m is the message being transmitted. Having a distinct network predicate for each message exchanged in a protocol specification, as done in [CDL$^+$99,DLMS99], is equivalent, but would obscure the translation in Section 5. In this paper, messages will consist of the class of terms freely generated from atomic messages (principal names, keys, nonces, etc.) by the operators of concatenation, denoted "$_,_$", and encryption, written "$\{_\}_$".

Role states: We first choose a set of *role identifiers* $\rho_1, \ldots, \rho_n$ for the different roles constituting the protocol. Then, for each role ρ, we have a finite family of *role state predicates* $\{\mathsf{A}_{\rho i}(\boldsymbol{m}) \mid i = 0 \ldots l_\rho\}$. They are intended to hold the internal state of a principal in role ρ during the successive steps of the protocol.

This scheme can immediately be generalized to express roles that can take conditional or non-deterministic actions (*e.g.* toss a coin to choose among two messages to send — useful for zero-knowledge proofs for examples — or respond in two different ways depending on the contents of an incoming message — useful for intrusion detection). We simply need to alter our naming convention for role states and rules (below) to take alternatives into account. Indeed, any partial ordering of the role state predicates will implement a *well-founded protocol theory*, as defined in [CDL$^+$99,DLMS99]. This paper will consider only linearly ordered role states, as the layer of technicality required to treat the general case would obscure the comparison with strands.

The additional predicate symbol I is needed to model the intruder's knowledge and its actions. It will be discussed at length in Section 2.3.

We represent each role ρ in a protocol by means of a single *role generation rule* and a finite number of *regular protocol execution rules*. The purpose of the former is to prepare for the execution of an instance of role ρ. It has the form

$$r_{\rho 0}: \ \boldsymbol{\pi}(\boldsymbol{x}) \rightarrow \mathsf{A}_{\rho 0}(\boldsymbol{x}), \boldsymbol{\pi}(\boldsymbol{x}).$$

where, here and in the rest of the paper, $\boldsymbol{\pi}(\boldsymbol{x})$ denotes a multiset of persistent atomic formulas that may mention variables among $\boldsymbol{x}$. Notice how persistent information is preserved. The execution rules describe the messages sent and expected by the principal

acting in this role. For $i = 0 \ldots l_\rho - 1$, we have a rule $r_{\rho i+1}$ of either of the following two forms:

$$\textbf{Send:} \quad \mathsf{A}_{\rho i}(\boldsymbol{x}) \quad \rightarrow \quad \exists \boldsymbol{n}.\, \mathsf{A}_{\rho i+1}(\boldsymbol{x}, \boldsymbol{n}),\, \mathsf{N}(m(\boldsymbol{x}, \boldsymbol{n}))$$

$$\textbf{Receive:} \; \mathsf{A}_{\rho i}(\boldsymbol{x}),\, \mathsf{N}(m(\boldsymbol{x}, \boldsymbol{y})) \quad \rightarrow \quad \mathsf{A}_{\rho i+1}(\boldsymbol{x}, \boldsymbol{y})$$

where $m(\boldsymbol{v})$ stands for a message pattern with variables among $\boldsymbol{v}$. In the first type of rules, we rely on the existential operator $\exists \boldsymbol{n}$ to model the ability of a principal to create nonces when sending a message. Situations where a principal both sends and receives a message, or sends multiple messages, can easily be expressed by these rules.

A protocol is specified as a set $\mathcal{R}$ of these *regular roles*. Every $\mathcal{R}$ constructed in this way is trivially a well-founded protocol theory [CDL$^+$99,DLMS99]. As an example, Figure 1 shows the encoding of the familiar simplified Needham-Schroeder public key protocol in the multiset rewriting notation. For the sake of readability, we omitted the keys in the persistent state predicates.

A *state* is a multiset of ground facts $\boldsymbol{S} = \boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{N}, \boldsymbol{I}$, where $\boldsymbol{A}$ is a multiset of role states $\mathsf{A}_{\rho i}(\boldsymbol{t})$, $\boldsymbol{N}$ is multiset of messages $\mathsf{N}(m)$ currently in transit, and $\boldsymbol{I}$ is a collection of predicates $\mathsf{I}(m)$ summarizing the intruder's knowledge. Notice in particular that the *initial state*, denoted $\boldsymbol{S}_0$, is just $\boldsymbol{\Pi}, \boldsymbol{I}_0$, where $\boldsymbol{I}_0$ contains the information (*e.g.* keys) initially known to the intruder.

The above *regular* protocol theories upgrade our original definition of (unqualified) protocol theories [CDL$^+$00] with the requirement that all the persistent information used during the execution of a role be accessed in its role generation rule: in [CDL$^+$99,DLMS99], $\boldsymbol{\pi}(\boldsymbol{z})$ can occur in execution rules as well. While the two definitions are equally acceptable in general, the regularity restriction brings us one step closer to the strand world, where all accessory values are chosen up-front. We discovered however that this is a slippery step as protocol theories cannot be regularized in general without losing transition sequences (see Section 4.1 for additional details on this issue). This is one more restriction that our multiset rewriting formalism shall abide by in order to set up a fair comparison with strand spaces.

2.3 Intruder Theory

The knowledge available at any instant to the intruder consists of the persistent information in $\boldsymbol{\Pi}$, of the unused portion of its initial knowledge $\boldsymbol{I}_0$ (*e.g.* the keys of dishonest principals), and of intercepted or inferred messages. We use the state predicate $\mathsf{I}(\_)$ to contain each piece of information known to the intruder. In particular, we represent the fact that the intruder "knows" m (a message, a key, etc.) as $\mathsf{I}(m)$. The overall knowledge of the intruder at any particular instant is indicated with $\boldsymbol{I}$. As mentioned above, we write $\boldsymbol{I}_0$ for the intruder's initial knowledge.

The capabilities of the intruder are modeled by the *standard intruder theory* $\mathcal{I}$ displayed in Figure 2. These rules are taken verbatim from [CDL$^+$99,DLMS99]. $\mathcal{I}$ implements the Dolev-Yao model [DY83,NS78] in our notation. For the sake of readability, we have grayed out the information produced by each rule. Observe that these rules display an overly conservative bookkeeping strategy for the known messages: knowledge is never discarded, but carried along as new messages are inferred.

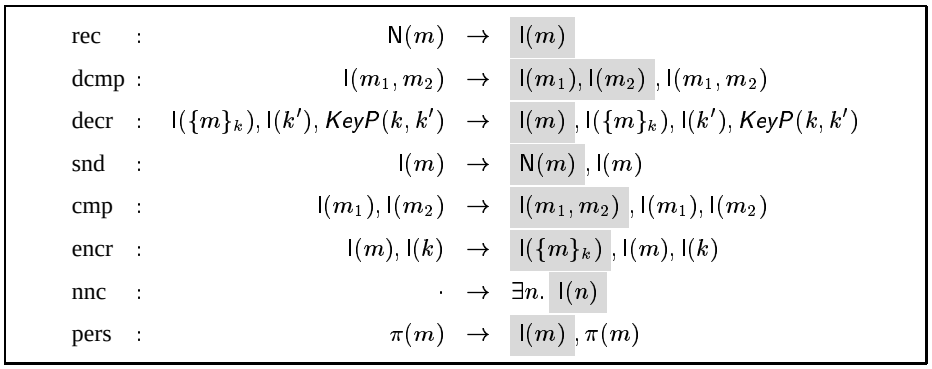


Fig. 2. The Standard Intruder Theory $\mathcal{I}$

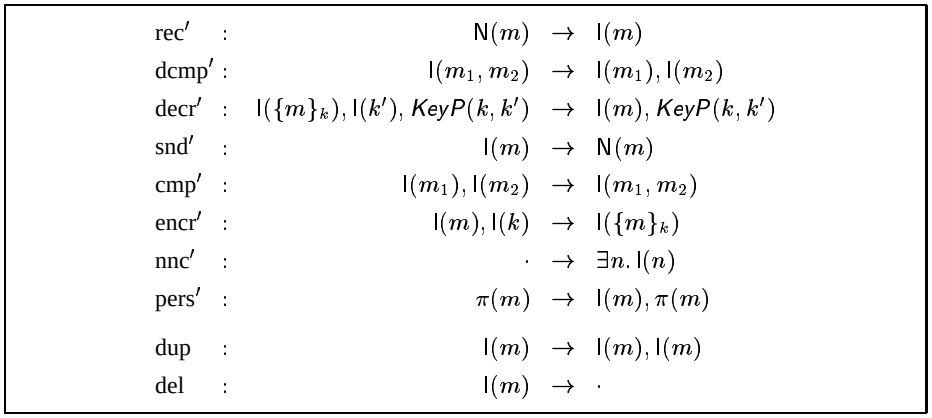


Fig. 3. The Modified Intruder Theory $\mathcal{I}'$

The intruder capabilities formalized in the strand model relies on a slightly different strategy for managing captured knowledge: inferring new information has the effect of deleting the data it was constructed from. Moreover, it can discard information. However, explicit duplication is possible. We express this behavior by the set of rules $\mathcal{I}'$ in Figure 3.

Clearly, our original intruder model $\mathcal{I}$ can easily be simulated by a systematic use of the duplication rule of $\mathcal{I}'$. Going in the other direction is slightly trickier as $\mathcal{I}$ never discards any information. The substantial equivalence of these two systems is summarized in the following result.

Property 1. Let $\mathcal{R}$ be an arbitrary protocol theory, and $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ two states.

- For every rule sequence $\boldsymbol{r}$ in $\mathcal{R}, \mathcal{I}$ such that $\boldsymbol{S}_1 \xrightarrow{\boldsymbol{r}}{}^*_{\mathcal{R},\mathcal{I}} \boldsymbol{S}_2$, there exists a rule sequence $\boldsymbol{r}'$ in $\mathcal{R}, \mathcal{I}'$ such that $\boldsymbol{S}_1 \xrightarrow{\boldsymbol{r}'}{}^*_{\mathcal{R},\mathcal{I}'} \boldsymbol{S}_2$.
- For every rule sequence $\boldsymbol{r}'$ in $\mathcal{R}, \mathcal{I}'$ such that $\boldsymbol{S}_1 \xrightarrow{\boldsymbol{r}'}{}^*_{\mathcal{R},\mathcal{I}'} \boldsymbol{S}_2$, there exist a rule sequence $\boldsymbol{r}$ in $\mathcal{R}, \mathcal{I}$ and an intruder state $\boldsymbol{I}'$ such that $\boldsymbol{S}_1 \xrightarrow{\boldsymbol{r}}{}^*_{\mathcal{R},\mathcal{I}} \boldsymbol{S}_2, \boldsymbol{I}'$.

Proof: The idea underlying the proof of the first statement is that every rule in $\mathcal{I}$ can be emulated by the corresponding rule in $\mathcal{I}'$ preceded by one or more applications of dup. Rule del is never used. The transition sequence $\boldsymbol{r}'$ is derived from $\boldsymbol{r}$ according to this strategy. A formal proof proceeds by induction on $\boldsymbol{r}$.

The proof of the second half of this property is based on the observation that rule dup can be emulated in $\mathcal{I}$ by applying snd and rec in succession. The additional intruder state $\boldsymbol{I}'$ consists of copies of intermediate information produced by the rules of $\mathcal{I}$ plus whatever data were explicitly discarded by using del. Again, this is formally proved by induction on $\boldsymbol{r}'$. □

3 Strand Constructions

We now define strands and related concepts. In order to simplify this task, we first recall some basic definitions from graph theory in Section 3.1. In Section 3.2, we adapt the definitions in [Son99], which is more concise than [FHG98]. In Section 3.3, we extend the strand formalism with a series of new concepts intended to ease the comparison with protocol theories. These extensions are of independent interest and therefore we discuss some of their properties.

3.1 Preliminary Definitions

A *directed graph* G is a pair $(S, \longrightarrow)$ where S is the set of *nodes* of G and $\longrightarrow \subseteq S \times S$ is the set of *edges* of G. We will generally write $\nu_1 \longrightarrow \nu_2$ for $(\nu_1, \nu_2) \in \longrightarrow$. A *directed labeled graph* G_L is a structure $(S, \longrightarrow, L, \Lambda)$ where $(S, \longrightarrow)$ is a directed graph, L is a set of *labels*, and $\Lambda : S \to L$ is a *labeling function* that associates a label to every node. In the sequel, all our graphs will be directed and labeled, but we will generally keep Λ implicit for simplicity. In particular, for $\nu \in S$ and $l \in L$, we will write "$\nu = l$" as an abbreviation of $\Lambda(\nu) = l$. However, for $\nu_1, \nu_2 \in S$, expressions of the form "$\nu_1 = \nu_2$" shall always refer to the nodes themselves, and not to their labels.

A graph $G = (S, \longrightarrow)$ is a *chain* if there is a total ordering $\nu_0, \nu_1, \ldots$ of the elements of S such that $\nu_i \longrightarrow \nu_j$ iff $j = i + 1$. A graph $G = (S, \longrightarrow)$ is a *disjoint union of chains* if $S = \bigcup_{i \in I} S_i$ and $\longrightarrow = \bigcup_{i \in I} \longrightarrow_i$ (for some set I) and $(S_i, \longrightarrow_i)$ are chains for each $i \in I$.

A *bipartite graph* is a structure $G = (S_1, S_2, \longrightarrow)$ such that S_1 and S_2 are disjoint, $(S_1 \cup S_2, \longrightarrow)$ is a graph, and if $\nu_1 \longrightarrow \nu_2$ then $\nu_1 \in S_1$ and $\nu_2 \in S_2$. Observe that all edges go from S_1 to S_2 (*i.e.* $\longrightarrow \subseteq S_1 \times S_2$). We say that $G = (S_1, S_2, \longrightarrow)$ is

- *functional* if $\longrightarrow$ is a partial function (*i.e.* if $\nu \longrightarrow \nu'_1$ and $\nu \longrightarrow \nu'_2$ imply $\nu'_1 = \nu'_2$).
- *injective* if $\longrightarrow$ is injective (*i.e.* if $\nu_1 \longrightarrow \nu'$ and $\nu_2 \longrightarrow \nu'$ imply $\nu_1 = \nu_2$).
- *surjective* if $\longrightarrow$ is surjective onto S_2 (*i.e.* for each $\nu' \in S_2$ there is $\nu \in S_1$ such that $\nu \longrightarrow \nu'$).

A *bi-graph* G is a structure $(S, \Longrightarrow, \longrightarrow)$ where both $(S, \Longrightarrow)$ and $(S, \longrightarrow)$ are graphs.

In the sequel, we will often rely on the natural adaptation of standard graph-theoretic notions (*e.g.* isomorphism) to labeled graphs and bi-graphs.

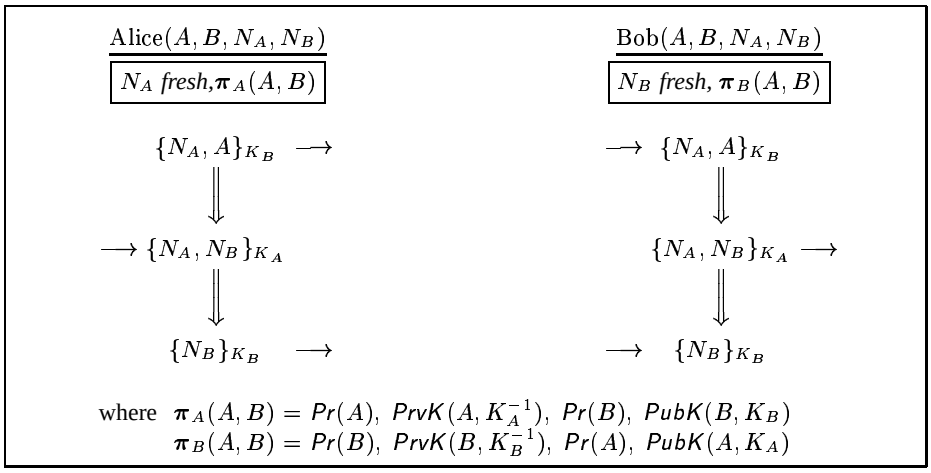


Fig. 4. Parametric Strand Specification of the Needham-Schroeder Protocol

3.2 Strands and Bundles

An *event* is a pair consisting of a message m and an indication of whether it has been sent ($+m$) or received ($-m$) [FHG98]. The set of all events will be denoted $\pm\mathcal{M}$.

A *strand* is a finite sequence of events, *i.e.* an element of $(\pm\mathcal{M})^*$. We indicate strands with the letter s, the length of a strand as $|s|$, and its i-th event as s_i (for $i = 1 \ldots |s|$). Observe that a strand s can be thought of as a chain graph $(S, \Longrightarrow)$ with labels over $\pm\mathcal{M}$, where $S = \{s_i : i = 1 \ldots |s|\}$ and $s_i \Longrightarrow s_j$ iff $j = i + 1$.

Slightly simplifying from [FHG98], a *strand space* is a set of strands with an additional relation ($\longrightarrow$) on the nodes. The only condition is that if $\nu_1 \longrightarrow \nu_2$, then $\nu_1 = +m$ and $\nu_2 = -m$ (for the same message m). Therefore, $\longrightarrow$ represents the transmission of the message m from the sender ν_1 to the receiver ν_2. Alternatively, a strand space can be viewed as a labeled bi-graph $\sigma = (S, \Longrightarrow, \longrightarrow)$ with labels over $\pm\mathcal{M}$, $\Longrightarrow \subseteq S \times S$, and $\longrightarrow \subseteq S^+ \times S^-$ where S^+ and S^- indicate the set of positively- and negatively-labeled nodes in S respectively, and the constraints discussed above: $(S, \Longrightarrow)$ is a disjoint union of chains, and if $\nu_1 \longrightarrow \nu_2$, then $\nu_1 = +m$ and $\nu_2 = -m$ for some message m.

A *bundle* is a strand space $\sigma = (S, \Longrightarrow, \longrightarrow)$ such that the bipartite graph $(S^+, S^-, \longrightarrow)$ is functional, injective, and surjective, and $(\Longrightarrow \cup \longrightarrow)$ is acyclic. In terms of protocols, the first three constraints imply that a message is sent to at most one recipient at a time, no message is received from more than one sender, and every received message has been sent, respectively. Dangling positive nodes correspond to messages in transit. We should point out that functionality is not required in [FHG98,Son99].

If we think in terms of protocols, a bundle represents a snapshot of the execution of a protocol (therefore a dynamic concept). As we will see, this comprises a current global state (what each principal and the intruder are up to, and the messages in transit), as well as a precise account of how this situation has been reached. Each role is expressed

as a strand in the current bundle. The intruder capabilities are themselves modeled as a fixed set of *penetrator strands*, which can be woven in a bundle. We skip the exact definitions [FHG98,Son99] as the construction we propose in the next sections will generalize them.

3.3 Extensions

We now introduce a few new concepts on top of these definitions. Besides contributing to the understanding of this formalism, they will ease the comparison with multiset rewriting specifications.

The notion of role is kept implicit in [FHG98] and rapidly introduced as the concept of *trace-type* in [Son99]. A *role* is nothing but a parametric strand: a strand where the messages may contain variables. An actual strand is obtained by instantiating all the variables in a parametric strand (or an initial segment of one) with persistent information and actual message pieces. For simplicity, we will not define nor consider constructions corresponding to arbitrary well-founded protocol theories (see Section 2 and [CDL$^+$99,DLMS99]).

A *parametric strand* for the role ρ may look as in Figure 5. The freshness of $\boldsymbol{n}$, *i.e.* the fact that the variables $\boldsymbol{n}$ should be instantiated with "new" constants that have not been used before, is expressed as a side condition. Using the terminology in [FHG98,Son99], the values $\boldsymbol{n}$ are *uniquely originated*. This is a slightly more verbose way of specifying freshness than our use of $\exists$ in the previous section, but it achieves the same effect. What we see as the main difference however is that freshness is presented as a meta-level comment in [FHG98,Son99], while we have it as an operator in our specification calculus. The relationship between variables are expressed in [Son99] using intuitive notation, *e.g.* k^{-1} for the inverse key of k, or k_A for the key of A. We formalize these relations by equipping ρ with the constraints $\boldsymbol{\pi}(\boldsymbol{x})$, that, without loss of generality, will be a set of persistent atomic formulas from Section 2, parameterized over $\boldsymbol{x}$.

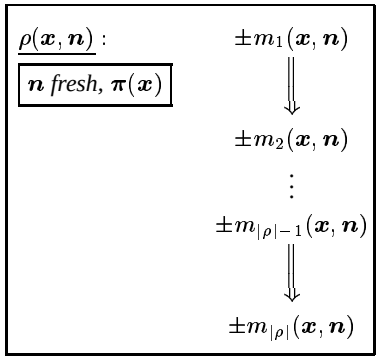


Fig. 5. A Parametric Strand

As in the case of transition systems, a *protocol* is given as a set of roles. The model of the intruder in the style of Dolev and Yao [DY83,NS78] is also specified as a set of parametric strands $\mathcal{P}(P_0)$ called *penetrator strands*, where P_0 is the intruder's initial knowledge (see Section 3.4 or [Son99] for a definition). As an example, Figure 4 shows how the Needham-Schroeder public key protocol is modeled using parametric strands, where we have used incoming and outgoing arrows instead of the tags $+$ and $-$ for readability.

As usual, a *substitution* is a finite tuple $\theta = (t_1/x_1, \ldots, t_n/x_n)$ of term-variable pairs t_i/x_i. The domain of θ is $\mathrm{dom}(\theta) = (x_1, \ldots, x_n)$, with each x_i distinct. All our substitution will be *ground*, by which we mean that none of the t_i's will contain any variable. We will rely on two types of substitutions: substitutions that replace variables

with distinct fresh constants that have not been previously encountered, and substitutions that map variables to previously used ground terms (not necessarily constants). We will use the letters ξ and θ, possibly subscripted, to denote them respectively. Given a parametric message m with variables in $\mathrm{dom}(\theta)$, we denote the application of θ to m as $m[\theta]$. Given substitutions $\theta_1, \ldots, \theta_n$, we write $m[\theta_1 \cdots \theta_n]$ for $(\ldots(m[\theta_1])\ldots)[\theta_n]$. We extend this notation to nodes, writing $\nu[\theta]$ and to (possibly partially instantiated) parametric strands, with the notation $\rho[\theta]$.

These definitions allow us to specialize the bundles we will be looking at: given a set of parametric strands $\mathcal{S}$, every strand in a bundle σ should be a fully instantiated initial prefix of a protocol (or penetrator) strand. We are interested in initial prefixes since a bundle is a snapshot of the execution of a protocol, and a particular role instance may be halfway through its execution. We then say that σ is a *bundle over* $\mathcal{S}$. We need to generalize strands constructions to admit strand spaces containing partially instantiated parametric strands. We call them *parametric strand spaces*. The bundles we will consider will however always be ground.

We will now give a few definitions needed to emulate the execution of a protocol with parametric strands. No such definitions can be found in the original description of strand constructions [FHG98,Son99], which focuses on analyzing protocol traces, not on specifying how to generate them.

First, observe that the network traffic in a bundle is expressed in terms of events and of the $\longrightarrow$ relation. The edges of $\longrightarrow$ represent past traffic: messages that have been sent and successfully received. The dangling positive nodes correspond to current traffic: messages in transit that have been sent, but not yet received. We will call these nodes the *fringe* of the bundle (or strand space). More formally, given a strand space $\sigma = (S, \Longrightarrow, \longrightarrow)$, its fringe is the set

$$\mathrm{Fr}(\sigma) = \{\nu : \nu \in S,\ \nu = +m,\ \text{and } \nexists \nu'.\, \nu \longrightarrow \nu'\}$$

Another component of the execution state of a protocol is a description of the actions that can legally take places in order to continue the execution. First, some technicalities. Let σ be a bundle over a set of parametric strands $\mathcal{S}$, a *completion* of σ is any strand space $\tilde{\sigma}$ that embeds σ as a subgraph, and that extends each incomplete strand in it with the omitted nodes and the relative $\Longrightarrow$-edges. A completion of σ may contain additional strands, possibly only partially instantiated. If s is a strand in σ and $\tilde{s}$ is its extension in $\tilde{\sigma}$, the sequence obtained by removing every event in s from $\tilde{s}$ is itself a (possibly empty) strand. We call it a *residual strand* and indicate it as $\tilde{s} \setminus s$. We then write $\tilde{\sigma} \setminus \sigma$ for the set of all residual strands of $\tilde{\sigma}$ with respect to σ, plus any strands that $\tilde{\sigma}$ may contain in addition to those in σ.

Given these preliminary definitions, a *configuration* over $\mathcal{S}$ is a pair of strand spaces $(\sigma, \sigma^\sharp)$ where σ is a bundle over $\mathcal{S}$, and $\sigma^\sharp$ is an extension of σ whose only additional $\longrightarrow$-edges originate in $\mathrm{Fr}(\sigma)$, cover all of $\mathrm{Fr}(\sigma)$, and point to $\sigma^\sharp \setminus \sigma$. Clearly, if $\sigma = (S, \Longrightarrow, \longrightarrow)$ and $\sigma^\sharp = (S^\sharp, \Longrightarrow^\sharp, \longrightarrow^\sharp)$, we have that $S \subseteq S^\sharp$, and $\Longrightarrow \subseteq \Longrightarrow^\sharp$, and finally $\longrightarrow \subseteq \longrightarrow^\sharp$.

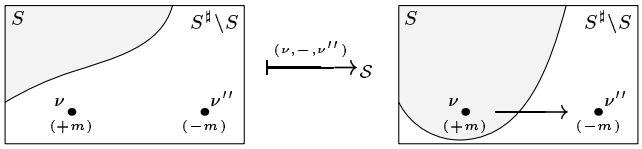
A one-step transition is what it takes to go from one bundle to the "next". There are two ways to make progress in the bundle world: extend a strand, or add a new one. Let us analyze them:

– *Extending a strand*: If the configuration at hand embeds a strand that is not fully contained in its bundle part, then we add the first missing node of the latter and the incoming $\Longrightarrow$-edge. If this node is positive, we add an $\longrightarrow$-arrow to a matching negative node. If it is negative, we must make sure that it has an incoming $\longrightarrow$-edge.
– *Creating a strand*: Alternatively, we can select a parametric strand and instantiate first its "fresh" data and then its other parameters. Were we to perform both instantiations at once, there would be no way to run protocols which exchange nonces, such as our example in Figure 4.

We will now formalize this notion. Let $(\sigma_1, \sigma_1^\sharp)$ and $(\sigma_2, \sigma_2^\sharp)$ be configurations over a set of parametric strands $\mathcal{S}$, with $\sigma_i = (S_i, \Longrightarrow_i, \longrightarrow_i)$ and $\sigma_i^\sharp = (S_i^\sharp, \Longrightarrow_i^\sharp, \longrightarrow_i^\sharp)$, for $i = 1, 2$. We say that $(\sigma_2, \sigma_2^\sharp)$ *immediately follows* $(\sigma_1, \sigma_1^\sharp)$ by means of move o, written $(\sigma_1, \sigma_1^\sharp) \overset{o}{\longmapsto}_{\mathcal{S}} (\sigma_2, \sigma_2^\sharp)$, if any of the following situations apply. An intuitive sense of what each case formalizes can be gained by looking at the pictorial abstraction to the right of each possibility. Here, ν, ν' and ν'' stand for nodes on fully instantiated strands, while ν_0 will generally be only partially instantiated.

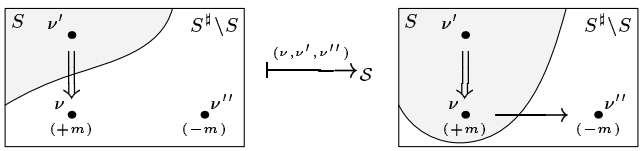
$\mathbf{S_0}$: There are nodes $\nu, \nu'' \in S_1^\sharp \setminus S_1$ such that $\nu = +m$, $\nu'' = -m$, no $\longrightarrow$-edge enters ν'', and no $\Longrightarrow$-arrow enters ν. Then,

– $S_2 = S_1 \cup \{\nu\}$,
$\Longrightarrow_2 = \Longrightarrow_1$,
$\longrightarrow_2 = \longrightarrow_1$;
– $S_2^\sharp = S_1^\sharp$,
$\Longrightarrow_2^\sharp = \Longrightarrow_1^\sharp$,
$\longrightarrow_2^\sharp = \longrightarrow_1^\sharp \cup \{(\nu, \nu'')\}$.



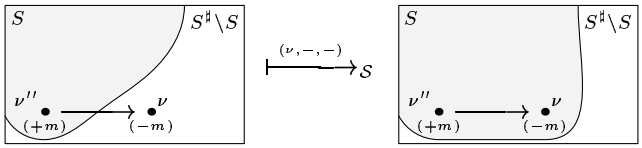
S: There are nodes $\nu, \nu'' \in S_1^\sharp \setminus S_1$ and $\nu' \in S_1$ such that $\nu = +m$, $\nu'' = -m$, no $\longrightarrow$-edge enters ν'', and $\nu' \Longrightarrow_1^\sharp \nu$. Then,

– $S_2 = S_1 \cup \{\nu\}$,
$\Longrightarrow_2 = \Longrightarrow_1 \cup \{(\nu', \nu)\}$,
$\longrightarrow_2 = \longrightarrow_1$;
– $S_2^\sharp = S_1^\sharp$,
$\Longrightarrow_2^\sharp = \Longrightarrow_1^\sharp$,
$\longrightarrow_2^\sharp = \longrightarrow_1^\sharp \cup \{(\nu, \nu'')\}$.



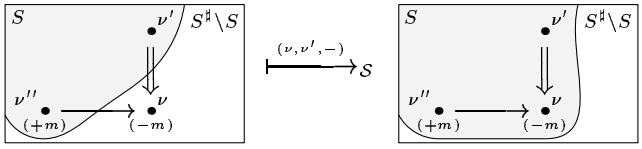
$\mathbf{R_0}$: There are nodes $\nu \in S_1^\sharp \setminus S_1$ and $\nu'' \in S_1$ such that $\nu = -m$, $\nu'' = +m$, $\nu'' \longrightarrow_1^\sharp \nu$, and no $\Longrightarrow$ enters ν. Then,

– $S_2 = S_1 \cup \{\nu\}$,
$\Longrightarrow_2 = \Longrightarrow_1$,
$\longrightarrow_2 = \longrightarrow_1 \cup \{(\nu'', \nu)\}$;
– $\sigma_2^\sharp = \sigma_1^\sharp$.



R: There are nodes $\nu \in S_1^\sharp \setminus S_1$ and $\nu', \nu'' \in S_1$ such that $\nu = -m$, $\nu'' = +m$, $\nu'' \longrightarrow_1^\sharp \nu$, and $\nu' \Longrightarrow_1^\sharp \nu$. Then,

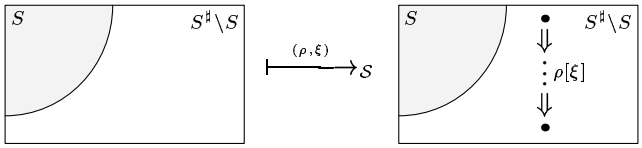
- $S_2 = S_1 \cup \{\nu\}$,
  $\Longrightarrow_2 = \Longrightarrow_1 \cup \{(\nu', \nu)\}$,
  $\longrightarrow_2 = \longrightarrow_1 \cup \{(\nu'', \nu)\}$;
- $\sigma_2^\sharp = \sigma_1^\sharp$.



$\mathbf{C_f}$: ρ is a parametric strand in $\mathcal{S}$ and ξ is a substitution for all its variables marked "fresh" with constants that appear nowhere in $(\sigma_1, \sigma_1^\sharp)$.

- $\sigma_2 = \sigma_1$;
- $\sigma_2^\sharp = \sigma_1^\sharp \cup \rho[\xi]$.

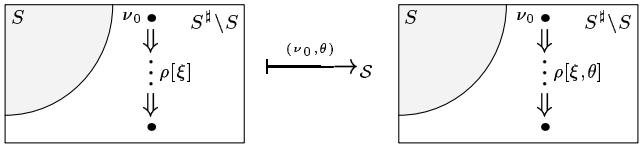
where, $\sigma \cup s$ is obtained by taking the union of the nodes and $\Longrightarrow$-edges of σ and s,



$\mathbf{C_i}$: $\rho[\xi]$ is a partially instantiated parametric strand in $\sigma_1^\sharp$ and θ is a ground substitution for the remaining variables. In particular, if $\rho[\xi]$ mentions constraints $\boldsymbol{\pi}$, then their instantiation should be compatible with the know persistent data, *i.e.* $\boldsymbol{\pi}[\theta] \subseteq \Pi$. Then,

- $\sigma_2 = \sigma_1$;
- $\sigma_2^\sharp = \sigma_1^\sharp - \rho[\xi] \cup \rho[\xi, \theta]$.

where, $\sigma - s$ is the subgraph of σ obtained by removing all nodes of s and their incident edges.



The *move* o that labels the transition arrow $\longmapsto_{\mathcal{S}}$ records the necessary information to reconstruct the transition uniquely. Given a configuration $(\sigma, \sigma^\sharp)$, a *move* for transitions of type $\mathbf{S_0}$, $\mathbf{S}$, $\mathbf{R_0}$, and $\mathbf{R}$ is a triple $o = (\nu, \bar{\nu}^p, \bar{\nu}^s)$ where ν is a node, $\bar{\nu}^p$ is

the parent node ν^p of ν according to the $\Longrightarrow$ relation (or "$-$" if ν is the first node of a chain — cases $\mathbf{S_0}$ and $\mathbf{R_0}$), and $\bar{\nu}^s$ is the recipient ν^s of the message that labels ν along the $\longrightarrow$ relation (if ν is positive, or "$-$" otherwise). For transitions of type $\mathbf{C_f}$ and $\mathbf{C_i}$, moves have the form (ρ, ξ) and (ν_0, θ) respectively, where ρ is the name of the chosen parametric strand, ν_0 is the first node of the partially instantiated strand $\rho[\xi]$, and ξ and θ are the instantiating substitutions.

A *multistep transition* amounts to chaining zero or more one-step transitions. This relation is obtained by taking the reflexive and transitive closure $\stackrel{\boldsymbol{o}}{\longmapsto}{}^*_{\mathcal{S}}$ of $\stackrel{\circ}{\longmapsto}_{\mathcal{S}}$, where $\boldsymbol{o}$ is the sequence of the component moves ("$\cdot$" if empty). $\boldsymbol{o}$ is a trace of the computation.

Observe that our definition of transition preserves configurations, *i.e.* if $(\sigma_1, \sigma_1^\sharp)$ is a configuration and $(\sigma_1, \sigma_1^\sharp) \stackrel{\circ}{\longmapsto}_{\mathcal{S}} (\sigma_2, \sigma_2^\sharp)$, then $(\sigma_2, \sigma_2^\sharp)$ is also a configuration. This property clearly extends to multistep transitions.

Property 2. Let $(\sigma_1, \sigma_1^\sharp)$ be a configuration. If $(\sigma_1, \sigma_1^\sharp) \stackrel{\boldsymbol{o}}{\longmapsto}{}^*_{\mathcal{S}} (\sigma_2, \sigma_2^\sharp)$, then $(\sigma_2, \sigma_2^\sharp)$ is a configuration.

Proof: By induction on the length of $\boldsymbol{o}$. □

3.4 Penetrator Strands

We now formalize the intruder model of [FHG98,Son99], which consists of patterns called *penetrator strands*, and of a set of messages P_0 expressing the intruder's initial knowledge. The corresponding parametric strands are shown in Figure 6, which includes a case to handle intruder-generated nonces. This possibility is missing from [FHG98,Son99], but the completion is straightforward. We also distinguished cases $M(m)$ and $M'(m)$, which are identified in [FHG98,Son99]. We refer to the collection of (parametric) penetrator strands in Figure 6 as $\mathcal{P}(P_0)$.

Several observations need to be made. First, the intruder specification underlying penetrator strands follows the Dolev-Yao model [DY83,NS78]. The parametric strands in Figure 6 are indeed closely related to the multiset rewriting intruder model $\mathcal{I}'$ above. A translation can be found in Sections 4.2 and 5.2 below, while a proof-sketch is embedded in the main results in Sections 4.2 and 5.2.

As a final remark, notice that the transition system specification distinguishes between messages transmitted on the network (identified by the predicate symbol N) and messages intercepted and manipulated by the intruder. Indeed, the predicate I implements a private database, a workshop for the fabrication of unauthorized messages, hidden from the honest principals of the system. No such distinction exists in the strand world. Therefore, it may seem that the intruder dismantles and puts together messages in the open, under the eyes of the other principals in the system. This is not the case: the privacy of the intruder is guaranteed by the fact that the $\longrightarrow$ relation is functional (see 3.2). Only the intruder can make use of intermediate results of penetrator manipulations since any other principal observing such messages would make them unavailable to the intruder (and it would not be an intermediate, but a final product of message forgery): since only one $\longrightarrow$-edge can leave a negative node and such an arrow is the only way to communicate (or observe somebody else's) data, the intruder could not access the message in this node for further processing.

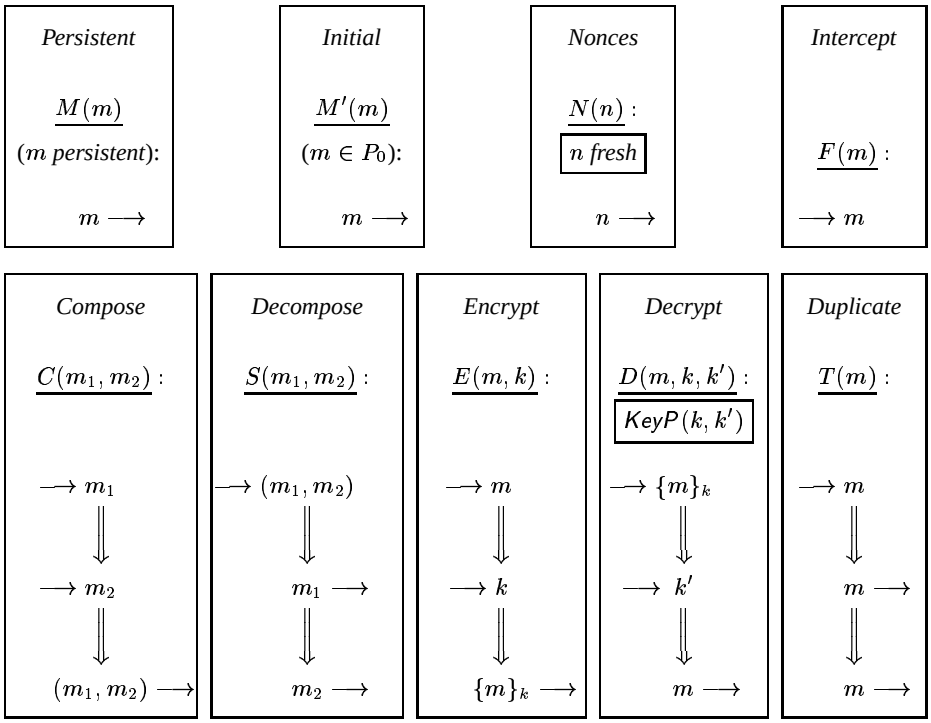


Fig. 6. The Penetrator Strands $\mathcal{P}$

The concepts and extensions we have just introduced set the basis for the translations between the multiset rewriting approach to security protocol specification and strand constructions. We describe the two directions of this translations in Sections 4 and 5, respectively.

4 From Multisets to Strands

The basic idea behind our translation will be to map a set of multiset rewrite rules specifying a role to a parametric strand. In particular, rules will correspond to nodes, and the role state predicates will be replaced by the backbone ($\Longrightarrow$) of the strand. In Section 4.1, we transform a regular protocol theory into an equivalent normal form. This transformation is novel and applies to a more general setting than the multiset rewriting specification of cryptoprotocols. In Section 4.2, we describe the translation proper and prove its correctness.

4.1 Normal Protocol Theories

We present two transformations which demonstrate that, without loss of generality, we can subsequently consider only normalized protocol theories. Their purpose is to re-

strict protocol theories so that they are closer to the strand model. Note that these transformations are used for mathematical convenience: non-normal, and even non-regular, protocol theories are often more perspicuous than their normalized counterparts.

Role generation rule: We subsume the role generation rule of every role ρ, *i.e.* the rule $r_{\rho 0} : \boldsymbol{\pi}(\boldsymbol{x}) \longrightarrow \mathsf{A}_{\rho 0}(\boldsymbol{x}), \boldsymbol{\pi}(\boldsymbol{x})$, into the first rule of ρ. For each of its two schematic forms:

$$r_{\rho 1} : \mathsf{A}_{\rho 0}(\boldsymbol{x}) \longrightarrow \exists \boldsymbol{n}.\, \mathsf{A}_{\rho 1}(\boldsymbol{x}, \boldsymbol{n}), \mathsf{N}(m(\boldsymbol{x}, \boldsymbol{n}))$$
$$r_{\rho 1} : \mathsf{A}_{\rho 0}(\boldsymbol{x}), \mathsf{N}(m(\boldsymbol{x}, \boldsymbol{y})) \longrightarrow \mathsf{A}_{\rho 1}(\boldsymbol{x}, \boldsymbol{y})$$

we obtain the following rules:

$$\ddot{r}_{\rho 1} : \boldsymbol{\pi}(\boldsymbol{x}) \longrightarrow \exists \boldsymbol{n}.\, \mathsf{A}_{\rho 1}(\boldsymbol{x}, \boldsymbol{n}), \mathsf{N}(m(\boldsymbol{x}, \boldsymbol{n})), \boldsymbol{\pi}(\boldsymbol{x})$$
$$\ddot{r}_{\rho 1} : \boldsymbol{\pi}(\boldsymbol{x}), \mathsf{N}(m(\boldsymbol{x}, \boldsymbol{y})) \longrightarrow \mathsf{A}_{\rho 1}(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{\pi}(\boldsymbol{x})$$

respectively. Observe that, by definition of role state predicate, the parameters $\boldsymbol{x}$ include the arguments of the elided $\mathsf{A}_{\rho 0}$ (as usual, $m(\boldsymbol{x})$ does not need to mention each variable in $\boldsymbol{x}$). This amounts to setting initial values in the first step of a role, rather than prior to any message exchange.
If $\mathcal{R}$ is a regular protocol theory, we will denote the effect of this transformation as $\ddot{\mathcal{R}}$. If $\boldsymbol{S}$ is a state, the transformed state $\ddot{\boldsymbol{S}}$ is obtained by dropping every mention of an initial role state $\mathsf{A}_{\rho 0}$ from $\boldsymbol{S}$. Clearly, $\ddot{\boldsymbol{S}}_0 = \boldsymbol{S}_0$ for any initial state $\boldsymbol{S}_0$. Similarly, a transition sequence $\boldsymbol{r}$ is mapped to a sequence $\ddot{\boldsymbol{r}}$ from which all the instances of rules for the form $r_{\rho 0}$ have been dropped, and the uses of $r_{\rho 1}$ have been replaced with $\ddot{r}_{\rho 1}$.
The above transformation is sound and complete as witnessed by the following result:

Lemma 1.
Let $\mathcal{R}$ be a regular protocol theory with initial state $\boldsymbol{S}_0$ and $\boldsymbol{S}$ a state. Then,

1. *If $\boldsymbol{S}_0 \xrightarrow{\boldsymbol{r}}{}^*_{\mathcal{R}} \boldsymbol{S}$, then $\boldsymbol{S}_0 \xrightarrow{\ddot{\boldsymbol{r}}}{}^*_{\ddot{\mathcal{R}}} \ddot{\boldsymbol{S}}$.*
2. *If $\boldsymbol{S}_0 \xrightarrow{\ddot{\boldsymbol{r}}}{}^*_{\ddot{\mathcal{R}}} \ddot{\boldsymbol{S}}$, then $\boldsymbol{S}_0 \xrightarrow{\boldsymbol{r}}{}^*_{\mathcal{R}} \boldsymbol{S}$.*

Proof: In both cases, the proof proceeds by induction on the length of the given transition sequences. □

Observe that applying this transformation and then "undoing" it as specified in the above lemma is not equivalent to the identical transformation: going in the reverse direction, we group occurrences of $r_{\rho 0}$ and $r_{\rho 1}$ together, and moreover we eliminate every isolated instance of $r_{\rho 0}$.

The original version of this paper [CDL+00] stated this lemma (or more precisely a result akin to the compounded Corollary 1 below) relative to general rather than regular protocol theories. This is incorrect: assume that $\boldsymbol{S}_0 \xrightarrow{\boldsymbol{r}}{}^*_{\mathcal{R}} \boldsymbol{S}_1$ thanks to the initialization rule $r_{\rho 0}$ of some role ρ. Assume also that the first message exchange rule $r_{\rho 1}$ of this role contains a persistent predicate which does not have any instantiation in $\boldsymbol{\Pi}$. The normal form of $r_{\rho 0}$ would then contain this constraint, making

it inapplicable to any state $\boldsymbol{S}_1$ would be mapped to. This scenario can clearly not occur when starting from a *regular* protocol theory since, by definition, all the accesses to persistent predicate are confined in the role instantiation rule.
An alternative way to correct the above error is to statically bind persistent information to any point in a protocol description where it is used. A realization of this idea by means of a strong typing infrastructure is at the basis of *MSR* [Cer01], a thorough redesign of the multiset rewriting formalism discussed in this paper. Besides being immune to that kind of errors, *MSR* is altogether a much better specification language as evidenced in [BCJS02].

Nonces: We further transform protocol theories so that all nonces generated by a role are preemptively chosen in the first rule of that role. We accomplish this by adding extra arguments to role state predicates, and pass the nonces generated in the first rule to subsequent uses through fresh variables in these predicates. Since roles are bounded, there are only a small finite number of nonces that need to be generated in an entire role. This transformation intuitively means that a participant should roll all her dice immediately, and look at them as needed later.
More formally, let ρ be the multiset rewriting specification of a role as from the previous transformation, and let $e_{\rho i}$ be the number of existentially quantified variables in rule $r_{\rho i}$, for $i = 1..|\rho|$. We map each role state predicate $\mathsf{A}_{\rho i}(\boldsymbol{x})$ in ρ to a predicate of the form

$$\bar{\mathsf{A}}_{\rho i}(\boldsymbol{x}, \boldsymbol{n}_{i+1}, \ldots, \boldsymbol{n}_{|\rho|})$$

where, for $j = i + 1..|\rho|$, there are exactly $e_{\rho j}$ elements in $\boldsymbol{n}_j$, and each of the added arguments is a distinct new variable.
We transform rules by replacing each state predicate $\mathsf{A}_{\rho i}$ with $\bar{\mathsf{A}}_{\rho i}$, and moving existential quantifiers to the first rule of the role. As a result, we are left with the following *normalized rules*:

Role generation rules:

- $\bar{r}_{\rho 1} : \boldsymbol{\pi}(\boldsymbol{x}) \longrightarrow \exists \boldsymbol{n}.\, \bar{\mathsf{A}}_{\rho 1}(\boldsymbol{x}, \boldsymbol{n}),\quad \mathsf{N}(m(\boldsymbol{x}, \boldsymbol{n})), \boldsymbol{\pi}(\boldsymbol{x})$
- $\bar{r}_{\rho 1} : \boldsymbol{\pi}(\boldsymbol{x}), \mathsf{N}(m(\boldsymbol{x}, \boldsymbol{y})) \longrightarrow \exists \boldsymbol{n}.\, \bar{\mathsf{A}}_{\rho 1}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{n}),\quad \boldsymbol{\pi}(\boldsymbol{x})$

Other rules:

- $\bar{r}_{\rho i+1} : \bar{\mathsf{A}}_{\rho i}(\boldsymbol{x}) \longrightarrow \bar{\mathsf{A}}_{\rho i+1}(\boldsymbol{x}),\quad \mathsf{N}(m(\boldsymbol{x}))$
- $\bar{r}_{\rho i+1} : \bar{\mathsf{A}}_{\rho i}(\boldsymbol{x}), \mathsf{N}(m(\boldsymbol{x}, \boldsymbol{y})) \longrightarrow \bar{\mathsf{A}}_{\rho i+1}(\boldsymbol{x}, \boldsymbol{y})$

where all the newly introduced variables in rule $\bar{r}_{\rho 1}$ are existentially quantified. Given a role ρ, we denote the normalized specification as $\bar{\rho}$. We write $\overline{\mathcal{R}}$ for the application of this transformation to a protocol theory $\mathcal{R}$.
In order to formally relate a regular protocol theory with its normalized form, we need to assess the effect of normalization on states. Given a ground predicate P in a state $\boldsymbol{S}$, we construct the open term $\overline{P}$ corresponding to the possible normalizations of P as follows:

$$\begin{cases} \overline{\mathsf{A}_{\rho i}(\boldsymbol{t})} = \bar{\mathsf{A}}_{\rho i}(\boldsymbol{t}, \boldsymbol{n}_i, \ldots, \boldsymbol{n}_{|\rho|}) & \text{where } \boldsymbol{n}_i, \ldots, \boldsymbol{n}_{|\rho|} \text{ consist of distinct variables} \\ \overline{P} \quad = P & \text{if } P \text{ is not a role state predicate} \end{cases}$$

It is easy to extend this definition to *open states*: if $\boldsymbol{S}$ is a state, we construct the open multiset $\overline{\boldsymbol{S}}$ representing all normalized states it is mapped to. $\overline{\boldsymbol{S}}$ is defined as

follows:

$$\overline{\boldsymbol{S}} = \wr \overline{P} : P \leftarrow \boldsymbol{S} \wr$$

where $\wr \ldots \wr$ is the multiset equivalent of the usual set notation $\{\ldots\}$, and $x \leftarrow M$ denotes multiplicity-conscious multiset membership. We shall choose different variables for each $\overline{P}$ in $\overline{\boldsymbol{S}}$. Observe that since the initial state $\boldsymbol{S}_0$ does not contain role state predicates, we have that $\overline{\boldsymbol{S}_0} = \boldsymbol{S}_0$.

The mapping between an open state $\overline{\boldsymbol{S}}$ and states that can be processed by transitions is done by means of substitutions ξ that map each variable in $\overline{\boldsymbol{S}}$ to a distinct constant that does not appear in $\boldsymbol{S}$. Observe that $\overline{\boldsymbol{S}}[\xi]$ is a (ground) state.

The definition of transition does not change, but we will denote a transition sequence that uses normalized rules as $\bar{\boldsymbol{r}}$ with the usual subscripts. We will shortly see how to normalize a transition sequence $\boldsymbol{r}$.

Given these various definitions, we are now in a position to prove that normalization preserves transitions. We have the following result.

Lemma 2.
Let $\ddot{\mathcal{R}}$ be a regular protocol theory that has been subjected to the role generation transformation in the first part of this section, $\boldsymbol{S}_0$ the initial state, and $\boldsymbol{S}$ a state. Let moreover ξ be an arbitrary substitution from the variables in $\overline{\boldsymbol{S}}$ to distinct unused constants. Then,

1. *If $\boldsymbol{S}_0 \xrightarrow{\boldsymbol{r}}{}^*_{\ddot{\mathcal{R}}} \boldsymbol{S}$, then $\boldsymbol{S}_0 \xrightarrow{\bar{\boldsymbol{r}}}{}^*_{\bar{\mathcal{R}}} \overline{\boldsymbol{S}}[\xi]$.*
2. *If $\boldsymbol{S}_0 \xrightarrow{\bar{\boldsymbol{r}}}{}^*_{\bar{\mathcal{R}}} \overline{\boldsymbol{S}}[\xi]$, then $\boldsymbol{S}_0 \xrightarrow{\boldsymbol{r}}{}^*_{\ddot{\mathcal{R}}} \boldsymbol{S}$.*

Proof: In both cases, the proof proceeds by induction on the length of the given transition sequences. We will examine them in turn. □

In the following we will start from a regular protocol theory $\mathcal{R}$ and apply these two transformations in sequence. For clarity reasons, we will generally write $\bar{\mathcal{R}}$ when $\ddot{\bar{\mathcal{R}}}$ would be appropriate. We extend this convention to roles and states.

The following corollary chains the above results together. It also considers protocols augmented with the standard intruder theory $\mathcal{I}$. It must be observed that the above transformations do not have any effect on $\mathcal{I}$.

Corollary 1.
Let $\mathcal{R}$ be a regular protocol theory, $\boldsymbol{S}_0$ the initial state, and $\boldsymbol{S}$ a state. Let moreover ξ be an arbitrary substitution from the variables in $\overline{\boldsymbol{S}}$ to distinct unused constants. Then,

1. *If $\boldsymbol{S}_0 \xrightarrow{\boldsymbol{r}}{}^*_{\mathcal{R},\mathcal{I}} \boldsymbol{S}$, then $\boldsymbol{S}_0 \xrightarrow{\bar{\boldsymbol{r}}}{}^*_{\bar{\mathcal{R}},\mathcal{I}} \overline{\boldsymbol{S}}[\xi]$.*
2. *If $\boldsymbol{S}_0 \xrightarrow{\bar{\boldsymbol{r}}}{}^*_{\bar{\mathcal{R}},\mathcal{I}} \overline{\boldsymbol{S}}[\xi]$, then $\boldsymbol{S}_0 \xrightarrow{\boldsymbol{r}}{}^*_{\mathcal{R},\mathcal{I}} \boldsymbol{S}$.*

Proof: This is a direct consequence of Lemmas 1 and 2 once we observe that the intruder rules never access the role state predicates of a principal. Therefore, the elision of the state predicate A_0 is invisible to the intruder. Similarly, the intruder cannot see nor take advantage of the fact that all existentials in a normal role have been instantiated up-front since they are safely stored in $\mathsf{A}_{\rho i}(\boldsymbol{x}, \boldsymbol{n}_i, \ldots, \boldsymbol{n}_{|\rho|})$ until they are made visible in a message. □

4.2 Translation

We are now in a position to translate protocol representations expressed in the multiset rewriting formalisms into strands. We first show how to map a general protocol theory into a set of parametric strands in Section 4.2, and then relate the intruder theory directly to the penetrators strands in Section 4.2. In Section 4.2, we prove that this translation preserve transitions after discussing how states are handled in Section 4.2.

From Protocol Theories to Parametric Strands To each normalized role specification $\bar{\rho}$, we associate a parametric strand $\ulcorner \bar{\rho} \urcorner$ of the following form

$$\underline{\rho(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{n})} \quad \boxed{\boldsymbol{n} \text{ fresh}, \boldsymbol{\pi}(\boldsymbol{x})}$$

where $\boldsymbol{n}$ are the existential variables mentioned in the first rule $\bar{r}_{\rho 1}$ of this role, $\boldsymbol{\pi}(\boldsymbol{x})$ are the persistent predicates accessed in this rule, and $\boldsymbol{y}$ are the other variables appearing in the role ($\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{n}$ appear therefore in its last role state predicate).

Next, we associate a parametric node $\nu_{\bar{r}_{\rho i}}$ with each rule $\bar{r}_{\rho i}$. The embedded message is the message appearing in the antecedent or the consequent of the rule, the distinction being accounted for by the associated action. More precisely, we have the following translation (where we have omitted the argument of the state predicates, the indication of the variables occurring in the message, persistent information, and the existential quantifiers appearing in the role generation rule):

$$\begin{aligned} \ulcorner \bar{\mathsf{A}}_{\rho i} \longrightarrow \bar{\mathsf{A}}_{\rho i+1}, \mathsf{N}(m) \urcorner &= +m \\ \ulcorner \bar{\mathsf{A}}_{\rho i}, \mathsf{N}(m) \longrightarrow \bar{\mathsf{A}}_{\rho i+1} \urcorner &= -m \end{aligned}$$

where $\ulcorner \_ \urcorner$ is our translation function.

Finally, we set the backbone of this parametric strand according to the order of the indices of the nodes (and rules):

$$\nu_{\bar{r}_{\rho i}} \Longrightarrow \nu_{\bar{r}_{\rho j}} \qquad \text{iff} \qquad j = i + 1.$$

In this way, we are identifying the role state predicates of the transition system specification with the $\Longrightarrow$-edges constituting the backbone of the corresponding parametric strand. Notice that the well-founded ordering over role state predicates is mapped onto the acyclicity of the $\Longrightarrow$-arrows of the strand constructions.

This completes our translation as far as roles, and therefore protocols, are concerned. Applying it to the Needham-Schroeder protocol yields exactly the parametric strand specification of Figure 4 presented in Section 3. Given a set of roles $\mathcal{R}$ in the transition system notation, we indicate the corresponding set of parametric strands as $\ulcorner \overline{\mathcal{R}} \urcorner$. We will give correctness results at the end of this section after showing how to translate global states.

From Intruder Theory to Penetrator Strands The introduction of the alternate intruder theory $\mathcal{I}'$ in Section 2.3 enables a trivial mapping to penetrator strands: we simply map every intruder rule to the corresponding penetrator strand, with the exception of rec' and snd', which do not have any correspondent. In symbols:

$$\begin{array}{rcl@{\qquad}rcl}
\ulcorner \mathrm{rec}'(m) \urcorner & = & \textit{none} & \ulcorner \mathrm{snd}'(m) \urcorner & = & \textit{none} \\
\ulcorner \mathrm{dcmp}'(m_1, m_2) \urcorner & = & S(m_1, m_2) & \ulcorner \mathrm{cmp}'(m_1, m_2) \urcorner & = & C(m_1, m_2) \\
\ulcorner \mathrm{decr}'(m, k) \urcorner & = & D(m, k) & \ulcorner \mathrm{encr}'(m, k) \urcorner & = & E(m, k) \\
\ulcorner \mathrm{nnc}'(n) \urcorner & = & N(n) & \ulcorner \mathrm{pers}'(m) \urcorner & = & M(m) \\
\ulcorner \mathrm{dup}(m) \urcorner & = & T(m) & \ulcorner \mathrm{del}(m) \urcorner & = & F(m)
\end{array}$$

where we have equipped the intruder rules with arguments in the obvious way. We also need to map the initial intruder knowledge $\boldsymbol{I}_0$ to a set P_0 of messages initially known to the intruder, to be processed by the penetrator strand M': $\ulcorner \boldsymbol{I}_0 \urcorner = \{m : \mathsf{I}(m) \in \boldsymbol{I}_0\}$. Every access to a message $\mathsf{I}(m)$ in $\boldsymbol{I}_0$ will be translated to an application of the penetrator strand $M'(m)$.

Relating States and Configurations In order to show that a transition system specification and its strand translation behave in the same way, we need to relate states and configurations. We do not need to give an exact mapping, since a configuration embeds a bundle expressing the execution up to the current point in fine detail. A state is instead a much simpler construction that does not contain any information about how it has been reached. Therefore, we will consider some properties that a configuration should have to be related to a state.

We say that a state $\boldsymbol{S} = \boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{N}(\boldsymbol{m}), \boldsymbol{I}(\boldsymbol{m}')$ is *compatible* with a strand configuration $(\sigma, \sigma^\sharp)$, written $\boldsymbol{S} \sim_{\mathcal{R}} (\sigma, \sigma^\sharp)$ relative to a protocol theory $\overline{\mathcal{R}}$, if the following conditions hold:

- $\mathrm{Fr}(\sigma) = \boldsymbol{m}, \boldsymbol{m}'$.
- Let $\bar{\mathsf{A}}_{\rho i}(\boldsymbol{t_\rho}, \boldsymbol{c_\rho})$ in $\boldsymbol{A}$ be the instantiation of the i-th role state predicate of a role $\bar{\rho}$ in $\overline{\mathcal{R}}$ with terms $\boldsymbol{t}_\rho$ and fresh nonces $\boldsymbol{c}_\rho$. Then,
  - $\sigma^\sharp$ contains a strand $s^\rho(\boldsymbol{c_\rho}, \boldsymbol{t_\rho})$, obtained by instantiating the strand $s^\rho = \ulcorner \bar{\rho} \urcorner$ with terms $\boldsymbol{t}_\rho$ and new constants $\boldsymbol{c}_\rho$.
  - σ contains an initial prefix of $s^\rho(\boldsymbol{t})$ whose last node has index i.

  Moreover every non-penetrator strand in $(\sigma, \sigma^\sharp)$ is obtained in this way.
- Every instance of a penetrator strand in $(\sigma, \sigma^\sharp)$ is completely contained in σ.

Intuitively, we want the state and the configuration to mention the same nonces, to have the same messages in transit (including the data currently processed by the intruder), to be executing corresponding role instances and have them be stopped at the same point.

Transition to Move Sequences Given these definitions, we can state the correctness result for our translation of transition systems into strand constructions. We shall start by limiting our attention to normal protocol theories together with the modified intruder theory introduced in Section 2.3.

Lemma 3.

Let $\overline{\mathcal{R}}$ be a normal protocol theory, $\boldsymbol{I}_0$ some initial intruder knowledge, and $\ulcorner \boldsymbol{I}_0 \urcorner$ its strand translation. If $\boldsymbol{\Pi}, \boldsymbol{I}_0 \xrightarrow{\boldsymbol{r}}{}^*_{\mathcal{I}', \mathcal{R}} \boldsymbol{S}$ is a normal multiset rewriting transition sequence over $\mathcal{I}', \overline{\mathcal{R}}$ from the empty state to state $\boldsymbol{S}$, then there is a configuration $(\sigma, \sigma^\sharp)$ and a sequence of moves $\boldsymbol{o}$ such that

$$(\cdot, \cdot) \xmapsto{\boldsymbol{o}}{}^*_{\mathcal{P}(\ulcorner \boldsymbol{I}_0 \urcorner), \ulcorner \overline{\mathcal{R}} \urcorner} (\sigma, \sigma^\sharp)$$

is a strand transition sequence from the empty configuration $(\cdot, \cdot)$ to $(\sigma, \sigma^\sharp)$, and $\boldsymbol{S} \sim_{\mathcal{R}} (\sigma, \sigma^\sharp)$, i.e. *$\boldsymbol{S}$ is compatible with $(\sigma, \sigma^\sharp)$.*

Proof: The proof proceeds by induction on $\boldsymbol{r}$. The base case is trivial. The inductive step does a case analysis on the last rule applied in $\boldsymbol{r}$. Intruder rules from $\mathcal{I}'$ are directly emulated by the corresponding penetrator strands, as defined in Section 4.2. The use of protocol rule $r_{\rho i}$ is emulated by a move involving the corresponding node in $\ulcorner \bar{\rho} \urcorner$. For each of these possibilities, we show that the corresponding move in the strand world is possible, and that it preserves the compatibility relation.

We omit formalizing this proof as it relies on exactly the same techniques as the proofs of previous results. □

We can now extend this result to any regular (not necessarily normal) theory together with the standard intruder model. We have the following theorem:

Theorem 1.

Let $\mathcal{R}$ a regular protocol theory and $\boldsymbol{I}_0$ be some initial intruder knowledge. For every regular multiset rewriting transition sequence $\boldsymbol{\Pi}, \boldsymbol{I}_0 \xrightarrow{\boldsymbol{r}}{}^*_{\mathcal{I}, \mathcal{R}} \boldsymbol{S}$ there is a configuration $(\sigma, \sigma^\sharp)$ and a sequence of moves $\boldsymbol{o}$ such that

$$(\cdot, \cdot) \xmapsto{\boldsymbol{o}}{}^*_{\mathcal{P}(\ulcorner I_0 \urcorner), \ulcorner \overline{\mathcal{R}} \urcorner} (\sigma, \sigma^\sharp)$$

is a strand transition sequence from the empty configuration $(\cdot, \cdot)$ to $(\sigma, \sigma^\sharp)$, and $\boldsymbol{S} \sim_{\mathcal{R}} (\sigma, \sigma^\sharp)$.

Proof: This is a simple corollary of the above lemma mediated by an application of Lemma 1 to move between regular and normal protocol theories, and Property 1 reconcile using the standard vs. the modified intruder theory. □

Observe that we cannot further relax the statement of this theorem to consider arbitrary (*i.e.* non-regular) protocol theories as regularization does not preserve transition sequences.

5 From Strands to Multisets

We will now show how to translate a set of parametric strands into a set of transition rules that preserve multistep transitions. Again, there is a slight mismatch between the two formalisms which is addressed in Section 5.1. This technical adjustment of our definition of strands will produce precisely the regular role transition rules we originally defined in Section 2. We describe the translation itself and prove it correct in Section 5.2.

5.1 Decorated Strands

In the previous section, we have observed and taken advantage of the fact that there is a close affinity between the rules in the transition system specification of a role and the nodes in a parametric strand. More precisely, a node together with the outgoing or incoming $\longrightarrow$-edge and an indication of what to do next corresponds to a transition. In transition systems, "what to do next" is specified through the role state predicates $\mathsf{A}_{\rho i}$; in strand constructions, by means of the $\Longrightarrow$-edges. Therefore, using the same intuition as in Section 4, we will translate $\Longrightarrow$-edges to state predicates. We need to equip these predicates with the appropriate arguments (while we were able to simply drop them in the inverse translation).

Before describing how to do so, we will address two other minor syntactic discrepancies: the absence of an (explicit) strand equivalent of the role generation rule $\boldsymbol{\pi}(\boldsymbol{x}) \longrightarrow \mathsf{A}_{\rho 0}(\boldsymbol{x}), \boldsymbol{\pi}(\boldsymbol{x})$, and the fact that, in the transition system specification of a role, there is a final state predicate that lingers in the global state no matter what other transitions take place.

Role Generation transition: We add a dummy initial node, say $\top$, to every strand, with no incoming or outgoing $\longrightarrow$-edges, and one outgoing $\Longrightarrow$-edge to the original first node of the strand.

Final state: Dually, we alter the definition of strands to contain a final node, say $\bot$, again without any incoming or outgoing $\longrightarrow$-edge, and with one incoming $\Longrightarrow$-arrow from the original last node of the strand.

This corresponds to redefining strands as strings drawn from the language $\top(\pm\mathcal{M})^*\bot$, rather than just $(\pm\mathcal{M})^*$. Notice that now every (proper) event has both a predecessor and a successor $\Longrightarrow$-edge.

With the addition of these auxiliary nodes, we can label each $\Longrightarrow$-arrow in a strand s with parameters $\boldsymbol{x}_s, \boldsymbol{n}_s$ ($\boldsymbol{n}_s$ marked *fresh*) and a predicate constant A_{si} with progressive indices i. In the case of parametric strands, we equip these labels with arguments drawn from its set of parameters as follows:

Initial arrow: $\top \Longrightarrow \nu$

This is the predicate A_{s0} labeling the $\Longrightarrow$-edge that links the added initial node $\top$ to the first node of the original strand. The arguments of A_{s0} will be $\boldsymbol{x}_s$.

Successor arrow to a positive node:

$$\ldots \overset{\mathsf{A}_{si}(\boldsymbol{x})}{\Longrightarrow} +m(\boldsymbol{x}, \boldsymbol{n}) \Longrightarrow \ldots$$

Let $\mathsf{A}_{si}(\boldsymbol{x})$ be the label of the incoming $\Longrightarrow$-edge of a positive node $\nu = +m(\boldsymbol{x}, \boldsymbol{n})$, where m mentions known variables among $\boldsymbol{x}$ and unused nonces $\boldsymbol{n}$ among $\boldsymbol{n}_s$. Then the outgoing $\Longrightarrow$-arrow of ν will have label $\mathsf{A}_{si+1}(\boldsymbol{x}, \boldsymbol{n})$.

Successor arrow to a negative node:

$$\ldots \overset{\mathsf{A}_{si}(\boldsymbol{x})}{\Longrightarrow} -m(\boldsymbol{x}, \boldsymbol{y}) \Longrightarrow \ldots$$

Let $\mathsf{A}_{si}(\boldsymbol{x})$ be the label of the incoming $\Longrightarrow$-edge of a positive node $\nu = -m(\boldsymbol{x}, \boldsymbol{y})$, where m mentions known variables among $\boldsymbol{x}$, and unseen data $\boldsymbol{y}$. Then, the outgoing $\Longrightarrow$-arrow of n will have label $\mathsf{A}_{si+1}(\boldsymbol{x}, \boldsymbol{y})$.

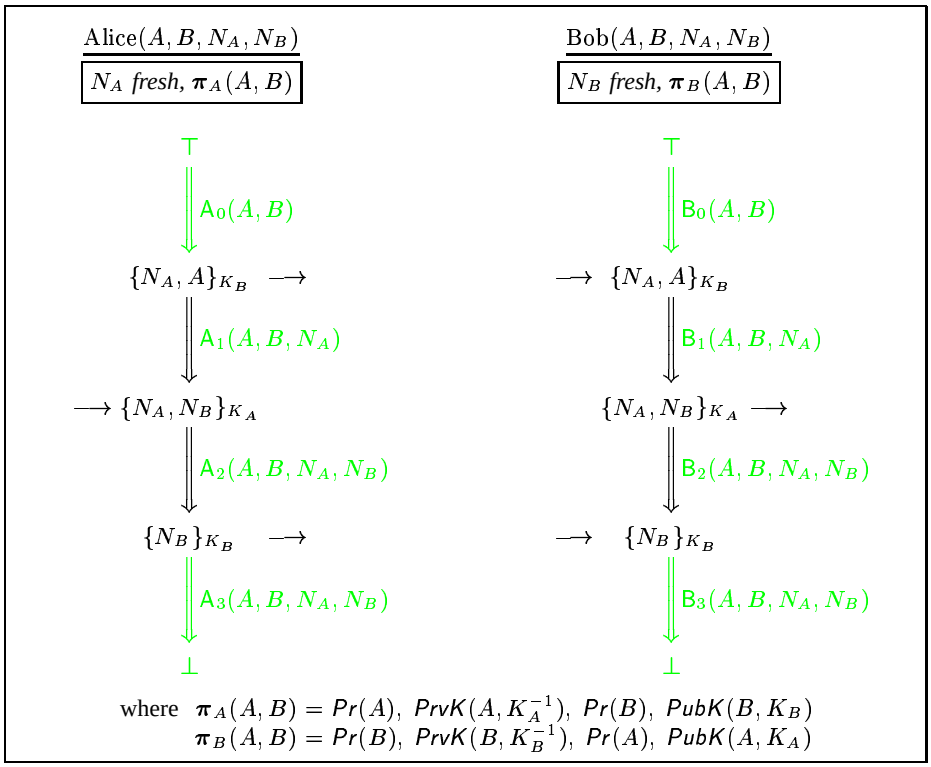


Fig. 7. Extended Strand Specification of the Needham-Schroeder Protocol

Given a parametric strand s, we denote the result of applying these transformations as $\bar{s}$. If $\mathcal{S}$ is a set of parametric strands specifying a protocol, we write $\overline{\mathcal{S}}$ for the transformed set. Applying this transformation to the Needham-Schroeder protocol yields the enhanced strand specification in Figure 7, where the additions have been grayed out.

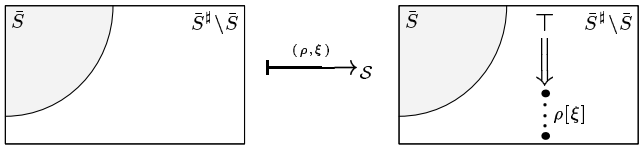
Since we have changed the syntax of a parametric strand, we need to upgrade its dynamics, originally presented in Section 2. First, an obvious alteration to the instantiation of a parametric strand: we apply the substitution to the labels of the $\Longrightarrow$-edges as well as to the messages embedded in the nodes. We carry on this change to the resulting bundles and configurations: every $\Longrightarrow$-edge between two nodes ν_1 and ν_2 now carries a label $\mathsf{A}_{si}(\boldsymbol{t})$. We indicate this as $\nu_1 \overset{\mathsf{A}_{si}(\boldsymbol{t})}{\Longrightarrow} \nu_2$ (or with its vertical equivalent). Notice that we erased this information in the reverse translation. Given a bundle σ and a configuration $(\sigma, \sigma^\sharp)$ relative to a set of parametric strands $\mathcal{S}$, we write $\bar{\sigma}$ and $(\bar{\sigma}, \bar{\sigma}^\sharp)$ for the corresponding entities relative to $\overline{\mathcal{S}}$.

The definition of one-step transition, in symbols $(\bar{\sigma}_1, \bar{\sigma}_1^\sharp) \overset{\circ}{\longmapsto}_{\overline{\mathcal{S}}} (\bar{\sigma}_2, \bar{\sigma}_2^\sharp)$, changes as follows:

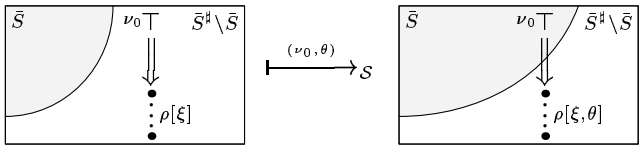
Extension of an existing strand: We proceed exactly as in Section 2, except for the fact that situations $\mathbf{S_0}$ and $\mathbf{R_0}$ in Section 3.3 do not apply.

Installation of a new strand:

We select a parametric strand ρ from $\overline{\mathcal{S}}$, instantiate it with a substitution ξ for its fresh variables and add the resulting strand $\rho[\xi]$ to $\bar{\sigma}_2^\sharp$. This corresponds to upgrading case $\mathbf{C_f}$ in Section 3.3 as outlined in the following figure. We do not formalize this transformation (call it $\mathbf{C_f}'$) it in full detail since it should be obvious how to obtain it.



Transition $\mathbf{C_i}$ is consequently upgraded to $\mathbf{C_i}'$ described in the following figure. Notice that we add the first node, $\top$, of $\rho[\xi,\theta]$ to $\bar{\sigma}_2$



As in the original case, multistep transitions are obtained by taking the reflexive and transitive closure of the above judgment.

This transformation is sound and complete with respect to our original system.

Lemma 4. *Let $\mathcal{S}$ be a set of parametric strands, and $(\sigma_1, \sigma_1^\sharp)$ and $(\sigma_2, \sigma_2^\sharp)$ two configurations on it. Then,*

$$(\sigma_1, \sigma_1^\sharp) \xmapsto{o}{}^*_{\mathcal{S}} (\sigma_2, \sigma_2^\sharp) \quad \textit{iff} \quad (\bar{\sigma}_1, \bar{\sigma}_1^\sharp) \xmapsto{\bar{o}}{}^*_{\overline{\mathcal{S}}} (\bar{\sigma}_2, \bar{\sigma}_2^\sharp)$$

where $\bar{\boldsymbol{o}}$ is obtained from $\boldsymbol{o}$ by extending the given transformation to traces.

Proof: In the forward direction, we add the labels as from the definition (they do not constrain the construction in any way); every use of transition $\mathbf{C_f}$ that introduces a new strand is mapped to $\mathbf{C_f}'$, which also installs the node $\top$. In the reverse direction, we simply forget about labels and extra nodes. Formally, both directions require a simple structural induction. □

5.2 Translation

Given the above definitions, we are in a position to propose an transition-preserving translation that maps strand representations of security protocols to the multiset rewriting formalism. We will proceed in stages: in Section 5.2 we concentrate on parametric strands, in Section 5.2 we relate the intruder models, in Section 5.2 we extract a notion of state from a configuration, and finally in Section 5.2 we prove the correctness of our translation.

From Parametric Strands to Roles We now present a translation of parametric strands to the coordinated sets of transition rules representing a role. Each node is mapped to a rule, the label of its incoming and outgoing $\Longrightarrow$-edge will be the state predicates

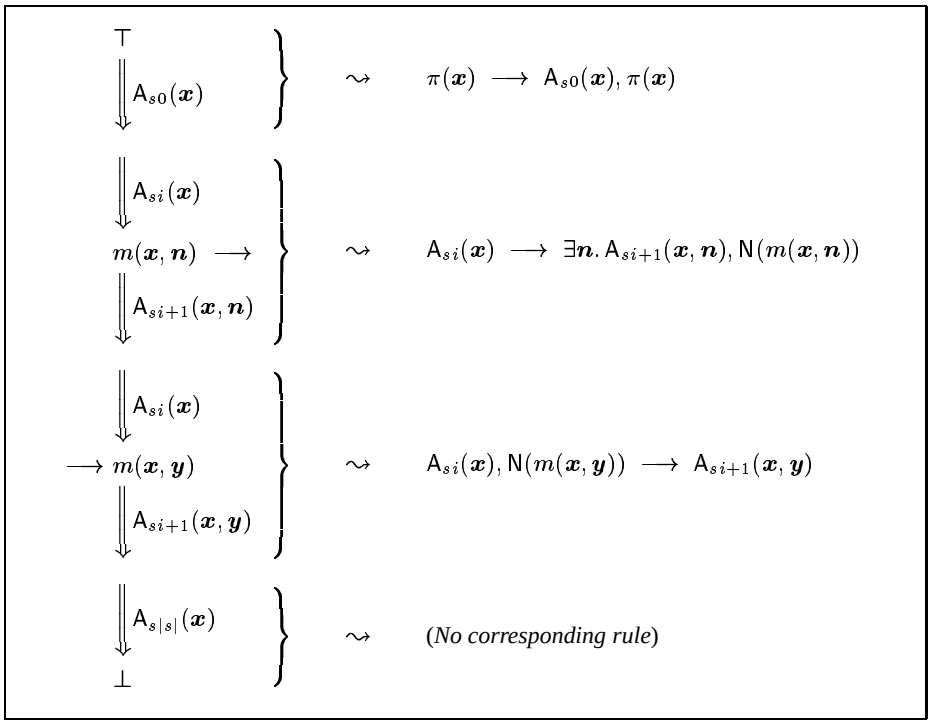


Fig. 8. Transforming Extended Strands to Multiset Rewriting Rules

in the antecedent and consequent, respectively, and the network message will be the message embedded in the node, its polarity dictating on which side of the arrow it should be appear. More formally, we have the translation displayed in Figure 8, where the parameters of the added state predicates are classified as in the above definition.

Given a set of (decorated) parametric strands $\overline{\mathcal{S}}$, we write $\ulcorner\overline{\mathcal{S}}\urcorner$ for the set of protocol rules resulting from this transformation. Observe that it yields regular rules. Applying this translation to the enhanced parametric strands representing the Needham-Schroeder protocol in Figure 7 produces exactly the original transition system specification given in Figure 1.

From Penetrator Strands to Intruder Theory The translation of the penetrator strands $\mathcal{P}(P_0)$ in Figure 6 is essentially the inverse of the mapping discussed in Section 4.2. Our target intruder model, in the multiset rewriting world, is $\mathcal{I}'$.

$$\begin{array}{rcl@{\qquad}rcl}
\ulcorner S(m_1, m_2)\urcorner & = & \mathrm{dcmp}'(m_1, m_2) & \ulcorner C(m_1, m_2)\urcorner & = & \mathrm{cmp}'(m_1, m_2) \\
\ulcorner D(m, k)\urcorner & = & \mathrm{decr}'(m, k) & \ulcorner E(m, k)\urcorner & = & \mathrm{encr}'(m, k) \\
\ulcorner N(n)\urcorner & = & \mathrm{nnc}'(n) & \ulcorner M(m)\urcorner & = & \mathrm{pers}'(m) \\
\ulcorner T(m)\urcorner & = & \mathrm{dup}(m) & \ulcorner F(m)\urcorner & = & \mathrm{del}(m) \\
\ulcorner M'(m)\urcorner & = & \textit{(see below)} & & &
\end{array}$$

where we have again equipped the intruder transition rules with the obvious arguments.

Notice that no penetrator strand is made to correspond to rules rec' or snd'. When translating transition sequences from the strand world to the transition system setting, we will insert these rules whenever a message sent by a principal's strand is received by a penetrator strand, and vice-versa, respectively. We map P_0 to a multiset $\boldsymbol{I}_0$ of messages initially known to the intruder in the multiset rewriting framework: $\ulcorner P_0 \urcorner = \wr \mathsf{I}(m) : m \in P_0 \wr$. Uses of $M'(m)$ with $m \in P_0$ are translated to accesses to $\mathsf{I}(m) \in \ulcorner P_0 \urcorner$, possibly preceded by an application of rule dup if $M'(m)$ is accessed more than once.

From Configurations to States Before we can show that the translation we just outlined preserves transition sequences, we need to extract a state from a configuration and show that steps between configurations are mapped to steps between the corresponding states.

Let $\mathcal{S}$ be a set of parametric strands, $\ulcorner \overline{\mathcal{S}} \urcorner$ its translation as a set of transition rules, and $(\sigma, \sigma^\sharp)$ a configuration over $\mathcal{S}, \mathcal{P}(P_0)$ where all penetrator strands have been completed. We define the *state associated with* $(\bar{\sigma}, \bar{\sigma}^\sharp)$, written $S_{\overline{\mathcal{S}}}(\bar{\sigma}, \bar{\sigma}^\sharp)$, as the state $\boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{N}, \boldsymbol{I}$ obtained as follows, where we write $\wr \ldots \wr$ for the multiset equivalent of the usual set notation $\{\ldots\}$:

- $\boldsymbol{N} = \wr \mathsf{N}(m) : \nu \in \mathrm{Fr}(\bar{\sigma}), \nu$ is not on a penetratorstrand, and ν has label $+m \wr$.
- $\boldsymbol{I} = \wr \mathsf{I}(m) : \nu \in \mathrm{Fr}(\bar{\sigma}), \nu$ is on a penetrator strand,and ν has label $+m \wr$.
- $\boldsymbol{A} = \wr \mathsf{A}_{si}(\boldsymbol{t}) : s_{i-1} \overset{\mathsf{A}_{si}(\boldsymbol{t})}{\Longrightarrow} s_i \in \bar{\sigma}^\sharp \setminus \bar{\sigma}$ and $s_{i-1} \in \mathrm{Fr}(\sigma) \wr$.

Intuitively, we collect the messages in transit coming from honest principal's strands in $\boldsymbol{N}$, the current knowledge of the intruder in $\boldsymbol{I}$, and the labels of the $\Longrightarrow$-edges at the boundary between $\bar{\sigma}^\sharp$ and $\bar{\sigma}$ as the multiset of role state predicates $\boldsymbol{A}$.

From Move to Transition Sequences Then, sequences of moves in the strand world and their translation as transition system steps are related as follows:

Theorem 2. *Let P_0 be some initial penetrator knowledge, and $\ulcorner P_0 \urcorner$ its multiset translation as defined in Section 5.2. Let $(\sigma_1, \sigma_1^\sharp)$ and $(\sigma_2, \sigma_2^\sharp)$ be two configurations on the penetrator strands $\mathcal{P}(P_0)$ and a set of parametric strands $\mathcal{S}$ such that all penetrator strands have been completed. For every multistep strand transition*

$$(\sigma_1, \sigma_1^\sharp) \overset{o}{\longmapsto}{}^*_{\mathcal{P}(P_0),\mathcal{S}} (\sigma_2, \sigma_2^\sharp),$$

and every $I_0' \subseteq \ulcorner P_0 \urcorner$, there exists a regular multiset transition sequence $\boldsymbol{r}$ such that

$$\ulcorner P_0 \urcorner, S_{\overline{\mathcal{S}}}(\bar{\sigma}_1, \bar{\sigma}_1^\sharp) \overset{\boldsymbol{r}}{\longrightarrow}{}^*_{\mathcal{I}', \ulcorner \overline{\mathcal{S}} \urcorner} S_{\overline{\mathcal{S}}}(\bar{\sigma}_2, \bar{\sigma}_2^\sharp), I_0'.$$

Proof: The proof of this result proceeds by induction on the structure of $\boldsymbol{o}$. The only non-obvious aspect is that, as observed in Section 5.2, we need to insert applications of the rule rec' when processing a message that flows from an honest principal's to a penetrator strands. We add uses of snd' in the dual case. □

Notice that we do not need to start from the empty configuration.

The mapping from strands to multiset rewriting we have just finished outlining, and the translation from multiset rewriting to strand constructions described in Section 4 are inverse of each other. We leave the proof of this fact to the interested reader.

6 Conclusions and Future Work

We have developed a formal connection between multiset rewriting [CDL+99,DLMS99] and strand constructions [FHG98,Son99]. The formalization of this unsurprising result required a number of unexpected adjustments to both frameworks. In particular, we equipped strands with a dynamic dimension by introducing a notion of transition that allows growing bundles from a set of parametric strands. This enabled us to relate the distinct notions of traces inherent to these formalisms: bundles and multiset rewrite sequences. On the other hand, we omitted the initialization phase of our multiset theories, since this phase has no counterpart in strand spaces.

This work can be applied to strengthen both formalisms. Our results imply that many multiset rewriting concepts and techniques devised over the years are likely to be relevant to the research on strands. The linear logic and rewriting logic foundations of multiset rewriting can thus be brought to bear on strand spaces as well. In addition, clean and intuitively appealing notions from strand spaces can be brought to multiset rewriting. For example, strand space bundles appear to be a better notion of computation trace than rewrite sequences, and therefore analogs could be fruitfully adopted in multiset rewrite systems. This has influenced the redesign of formalism presented here as the *MSR* security protocols specification language [Cer01] and fuels current research in the area. Finally, our work suggests extending strand spaces by embedding an explicit form of initialization, and refining the notion of initialization theories of multiset rewriting.

This paper can also be viewed as another step in a larger program of demonstrating connections between formalisms: the interoperation of logical systems can lead to improvements in the newly connected systems, but also lead to a deeper understanding of the entire problem domain. In this case, we have gained insight into the Dolev-Yao model of cryptoprotocols. Further connections to other formalisms including state-transition systems and linear logic [CDKS00] can improve the situation further. In fact, we are currently investigating properties of the representation of both strand and multiset rewriting constructions as process specification languages such as colored Petri nets and process algebras.

Acknowledgments

We would like to thank Joshua Guttman, Javier Thayer Fábrega, Jonathan Herzog, and Al Maneki for the stimulating discussions about strands. We are also indebted to Sylvan Pinsky for his encouragements to write down our ideas about the relationship between strand construction and our protocol theories. Finally, this work profitted from fruitful discussions with Jon Millen, Cathy Meadows, and Paul Syverson.

References

[BCJS02] Frederic Butler, Iliano Cervesato, Aaron D. Jaggard, and Andre Scedrov. A Formal Analysis of Some Properties of Kerberos 5 Using MSR. In *Fifteenth Computer Security Foundations Workshop — CSFW-15*, pages 175–190, Cape Breton, NS, Canada, June 2002. IEEE Computer Society Press.

[CDKS00] Iliano Cervesato, Nancy Durgin, Max I. Kanovich, and Andre Scedrov. Interpreting Strands in Linear Logic. In H. Veith, N. Heintze, and E. Clark, editors, *2000 Workshop on Formal Methods and Computer Security — FMCS'00*, Chicago, IL, July 2000.

[CDL+99] Iliano Cervesato, Nancy A. Durgin, Patrick D. Lincoln, John C. Mitchell, and Andre Scedrov. A meta-notation for protocol analysis. In P. Syverson, editor, *Proceedings of the 12th IEEE Computer Security Foundations Workshop — CSFW'99*, pages 55–69, Mordano, Italy, June 1999. IEEE Computer Society Press.

[CDL+00] Iliano Cervesato, Nancy A. Durgin, Patrick D. Lincoln, John C. Mitchell, and Andre Scedrov. Relating strands and multiset rewriting for security protocol analysis. In P. Syverson, editor, *13th IEEE Computer Security Foundations Workshop — CSFW'00*, pages 35–51, Cambrige, UK, 3–5 July 2000. IEEE Computer Society Press.

[Cer01] Iliano Cervesato. Typed MSR: Syntax and Examples. In V.I. Gorodetski, V.A. Skormin, and L.J. Popyack, editors, *First International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security — MMM'01*, pages 159–177, St. Petersburg, Russia, May 2001. Springer-Verlag LNCS 2052.

[DLMS99] Nancy Durgin, Patrick Lincoln, John Mitchell, and Andre Scedrov. Undecidability of bounded security protocols. In N. Heintze and E. Clarke, editors, *Proceedings of the Workshop on Formal Methods and Security Protocols — FMSP*, Trento, Italy, July 1999. Extended version at `ftp://ftp.cis.upenn.edu/pub/papers/scedrov/msr-long.ps`.

[DM99] Grit Denker and Jonathan K. Millen. CAPSL Intermediate Language. In N. Heintze and E. Clarke, editors, *Proceedings of the Workshop on Formal Methods and Security Protocols — FMSP*, Trento, Italy, July 1999.

[DY83] Danny Dolev and Andrew C. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 2(29):198–208, 1983.

[FHG98] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pages 160–171, Oakland, CA, May 1998. IEEE Computer Society Press.

[FHG99] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Mixed strand spaces. In P. Syverson, editor, *Proceedings of the 12th IEEE Computer Security Foundations Workshop — CSFW'99*, pages 72–82, Mordano, Italy, June 1999. IEEE Computer Society Press.

[Man99] A. Maneki. Honest functions and their application to the analysis of cryptographic protocols. In P. Syverson, editor, *Proceedings of the 12th IEEE Computer Security Foundations Workshop — CSFW'99*, pages 83–89, Mordano, Italy, June 1999. IEEE Computer Society Press.

[NS78] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[Son99] Dawn Song. Athena: a new efficient automatic checker for security protocol analysis. In *Proceedings of the Twelth IEEE Computer Security Foundations Workshop*, pages 192–202, Mordano, Italy, June 1999. IEEE Computer Society Press.