Relating Strands and Multiset Rewriting for Security Protocol Analysis*

I. Cervesato
ITT Industries
iliano@itd.nrl.navy.mil

N. Durgin, J. Mitchell
Stanford University
{nad, jcm}@cs.stanford.edu

P. Lincoln
SRI International
lincoln@csl.sri.com

A. Scedrov
U. of Pennsylvania
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 [6] and multiset rewriting with existential quantification [2, 5]. 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.

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 [9] and from a simplified model presented in [4]. 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 [6, 7, 8]. 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 [10].

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 [2, 5]. 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 [3]. As presented in [2, 5], 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

*Partially supported by DoD MURI "Semantic Consistency in Information Exchange" as ONR Grant N00014-97-1-0505, by NSF Grants CCR-9509931, CCR-9629754 and CCR-9800785, and by NRL under contract N0014-96-D2024 to various authors.

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. 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.

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.

2.1 Multiset Rewriting

A *multiset* 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 M and N by taking their *multiset union*, denoted "M, N". The elements we will consider here will be first-order atomic formulas $A(\vec{t})$ over some signature.

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

$$r : \ F(\vec{x}) \longrightarrow \exists \vec{n}.\, G(\vec{x}, \vec{n})$$

where r is a label and $\exists \vec{n}$ indicates that the constants $\vec{n}$ 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 M, a rule $r : \ F(\vec{x}) \longrightarrow \exists \vec{n}.\, G(\vec{x}, \vec{n})$ is *applicable* if $M = F(\vec{t}), M'$, for terms $\vec{t}$. Then, *applying* r to M yields the multiset $N = G(\vec{t}, \vec{c}), M'$ where the constants $\vec{c}$ are fresh (in particular, they do not appear in M), $\vec{x}$ and $\vec{n}$ have been instantiated with $\vec{t}$ and $\vec{c}$ respectively, and the facts $F(\vec{t})$ in M have been replaced with $G(\vec{t}, \vec{c})$ to produce N. 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:

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

respectively. The labels r and $\vec{r}$ identify which rule(s) have been applied and the terms $\vec{t}$ used to instantiate $\vec{x}$. Thus, $\vec{r}$ acts as a complete trace of the execution.

2.2 Protocol Theories

We model protocols by means of specifically tailored multiset rewriting systems. We present here a simplified version of the model introduced in [2, 5]. This will prove sufficient for a comparison with the strand formalism. However, we refer the interested reader to these presentations and to [1] for a more detailed account. 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 [2, 5], 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 Π be the resulting set of ground facts (constraints on the initialization theory prevent Π from containing duplicates [2, 5]). Strand constructions assume instead that the persistent information is given as a set. We reconcile the two approaches by dropping the explicit initialization phase of [2, 5] and assuming Π given. We will allow individual rules to query Π (but not to modify it).

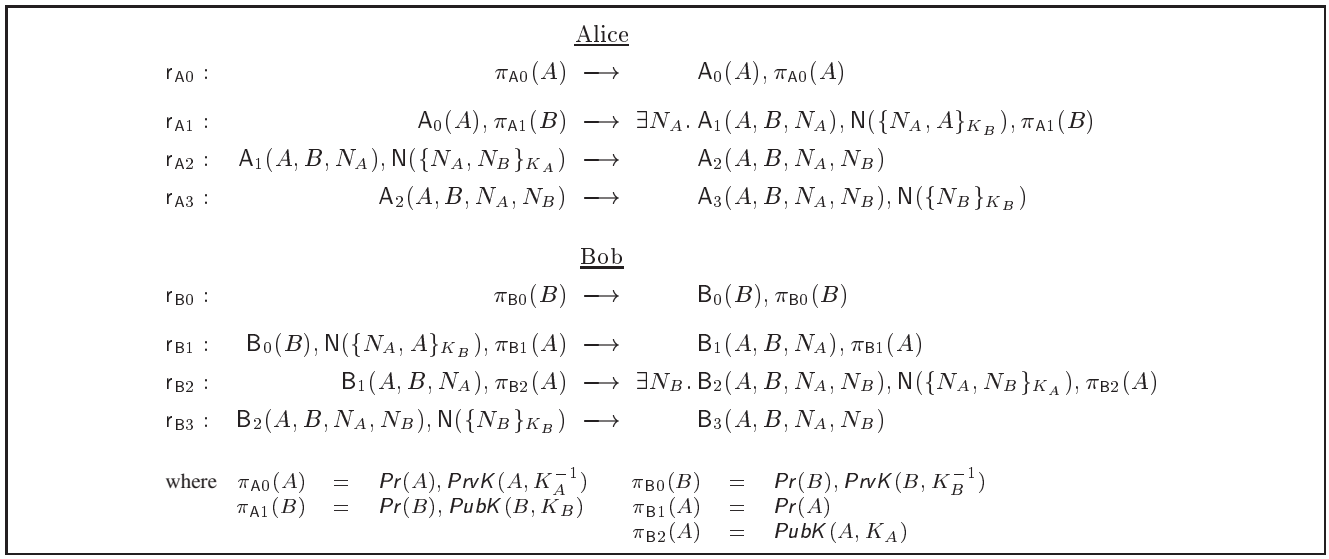


Figure 1. Multiset Rewriting Specification of the Needham-Schroeder Protocol

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 [2, 5], is equivalent, but would obscure the translation in Section 5.

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* $\{\mathsf{A}_{\rho i}(\vec{m}) \mid i = 0 \ldots l_\rho\}$. 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 [2, 5]. 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.

An additional predicate symbol (I) is needed to model the intruder's knowledge (see Appendix A).

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

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

where, here and in the rest of the paper, $\pi(\vec{x})$ denotes a multiset of persistent atomic formulas that may mention variables among $\vec{x}$. 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:

send:
$$\mathsf{A}_{\rho i}(\vec{x}), \pi(\vec{x}, \vec{z}) \longrightarrow \exists \vec{n}.\, \mathsf{A}_{\rho i+1}(\vec{x}, \vec{z}, \vec{n}), \mathsf{N}(m(\vec{x}, \vec{z}, \vec{n})), \pi(\vec{x}, \vec{z})$$

receive:
$$\mathsf{A}_{\rho i}(\vec{x}), \mathsf{N}(m(\vec{x}, \vec{y})), \pi(\vec{x}, \vec{y}, \vec{z}) \longrightarrow \mathsf{A}_{\rho i+1}(\vec{x}, \vec{y}, \vec{z}), \pi(\vec{x}, \vec{y}, \vec{z})$$

where $m(\vec{v})$ stands for a message pattern with variables among $\vec{v}$. In the first type of rules, we rely on the existential operator $\exists \vec{n}$ to model the ability of a principal to create nonces when sending a message. This principal can also include some persistent data $\vec{z}$ (*e.g.* the name and public key of an interlocutor), possibly related to information it already possesses ($\vec{x}$). In the second rule template, the principal should be able to access persistent information $\vec{z}$ related to data in the received message $\vec{y}$ (*e.g.* the sender's public key) or previously known information $\vec{x}$. Situations where a principal both sends and receive a message, or sends multiple messages, can easily be expressed by these rules.

A protocol is specified as a set $\mathcal{R}$ of such roles. Every $\mathcal{R}$ constructed in this way is trivially a well-founded protocol theory [2, 5]. As an example, Figure 1 shows the encoding

of the familiar simplified Needham-Schroeder public key protocol in the multiset rewriting notation. We used *Alice* and *Bob* as nicknames for the initiator and responder, respectively. For the sake of readability, we omitted the keys in the persistent state predicates.

The behavior of the intruder according to the Dolev-Yao model [4, 9] is similarly specified as a set of rewrite rules [1, 2]. We describe it in Appendix A. We will refer to them as $\mathcal{I}$. A state is then a multiset of ground facts $S = \Pi, A, N, I$, where A is a multiset of role states $\mathsf{A}_{\rho i}(\vec{t})$, N is multiset of messages $\mathsf{N}(m)$ currently in transit, and I summarizes the intruder's knowledge $\mathsf{I}(m)$. Notice in particular that the initial state is just Π, I_0, where I_0 contains the information (*e.g.* keys) initially known to the intruder.

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 [10], which is more concise than [6]. 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, 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, 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 $\nu_2 \in S_2$ (*i.e.* $\longrightarrow \subseteq S_1 \times S_2$). Observe that all edges go from S_1 to 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 $\nu_1 = \nu_2$).

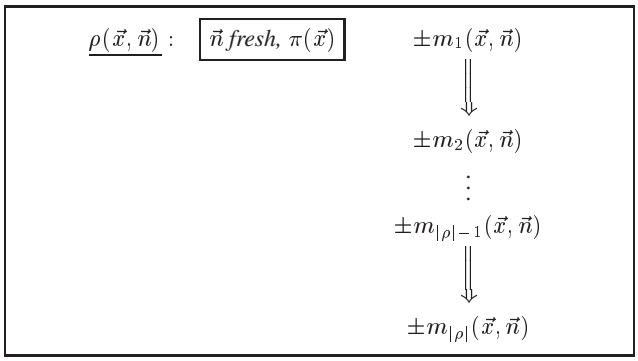


Figure 2. A Parametric Strand

- *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.

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 $(-m)$ [6]. 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 [6], 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

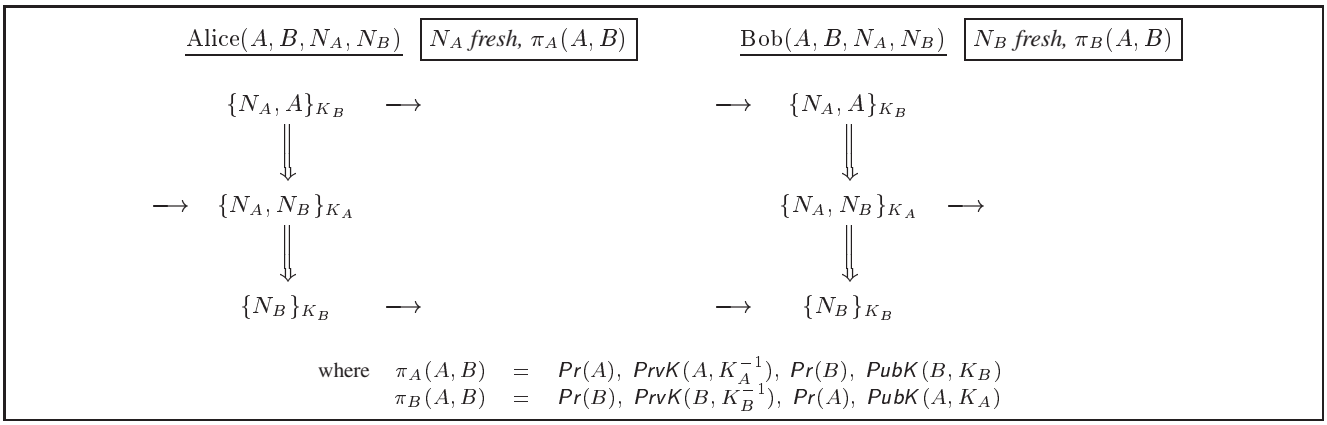


Figure 3. Parametric Strand Specification of the Needham-Schroeder Protocol

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 [6, 10].

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.

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 [6] and rapidly introduced as the concept of *trace-type* in [10]. 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 [2, 5]).

A *parametric strand* for the role ρ may look as in Figure 2. The freshness of $\vec{n}$, *i.e.* the fact that the variables $\vec{n}$ should be instantiated with "new" constants that have not been used before, is expressed as a side condition. Using the terminology in [6, 10], the values $\vec{n}$ 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 [6, 10], while we have it as an operator in our specification calculus. The relationship between variables are expressed in [10] 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 $\pi(\vec{x})$, that, without loss of generality, will be a set of persistent atomic formulas from Section 2, parameterized over $\vec{x}$.

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 [4, 9] 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 Appendix A or [10] for a definition, and [1] for an analysis). As an example, Figure 3 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.

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 an initial prefix of an instantiated 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 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 [6, 10], 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

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

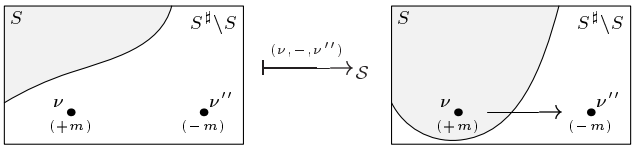
Another component of the execution state of a protocol is a description of the actions that can legally take places in order to continue the execution. First, some technicalities. Let σ be a bundle over a set of parametric strands $\mathcal{S}$, a *completion* of σ is any strand space $\tilde{\sigma}$ that embeds σ as a subgraph, and that extends each incomplete strand in it with the omitted nodes and the relative $\Longrightarrow$-edges. 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.

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:

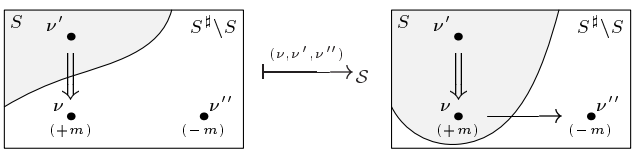
- *Extending a strand*: If the configuration at hand embeds a strand that is not fully contained in its bundle part, then we add the first missing node of the latter and the incoming $\Longrightarrow$-edge. If this node is positive, we add an $\longrightarrow$-arrow to a matching negative node. If it 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 3.

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 preceding each possibility. Here, ν, ν' and ν'' stand for nodes on fully instantiated strands.



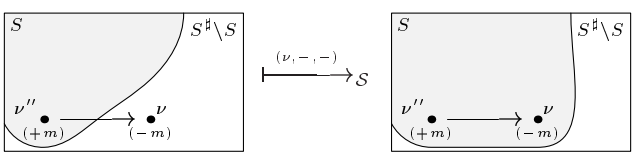
$\mathbf{S_0}$: There are nodes $\nu, \nu'' \in S_1^\sharp \setminus S_1$ such that $\nu = +m$, $\nu'' = -m$, no $\longrightarrow$-edge enters ν'', and no $\Longrightarrow$-arrow enters ν. Then,

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



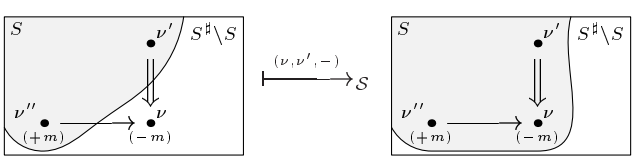
S: There are nodes $\nu, \nu'' \in S_1^\sharp \setminus S_1$ and $\nu' \in S_1$ such that $\nu = +m$, $\nu'' = -m$, no $\longrightarrow$-edge enters ν'', and $\nu' \Longrightarrow_1^\sharp \nu$. Then,

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



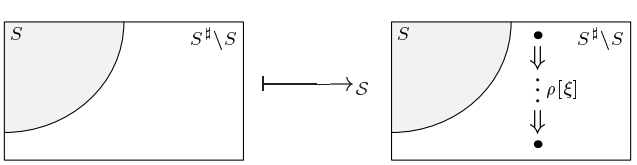
$\mathbf{R_0}$: There are nodes $\nu \in S_1^\sharp \setminus S_1$ and $\nu'' \in S_1$ such that $\nu = -m$, $\nu'' = +m$, $\nu'' \longrightarrow_1^\sharp \nu$, and no $\Longrightarrow$ enters ν. Then,

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



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,

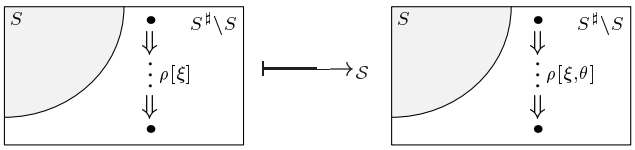
- $S_2 = S_1 \cup \{\nu\}$, $\Longrightarrow_2 = \Longrightarrow_1 \cup \{(\nu', \nu)\}$, $\longrightarrow_2 = \longrightarrow_1 \cup \{(\nu'', \nu)\}$;
- $\sigma_2^\sharp = \sigma_1^\sharp$.



$\mathbf{C_f}$: 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 substitution for the remaining variables. In particular, if $\rho[\xi]$ mentions constraints π, then their instantiation should be compatible with the know persistent data, *i.e.* $\pi[\theta] \subseteq \Pi$. 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 up to the creation of new strands. 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 node ν^s sending the message that labels ν along the $\longrightarrow$ relation (if ν is negative, or "$-$" otherwise). In order to simplify our analysis, we shall assume that transitions of type $\mathbf{C_f}$ and $\mathbf{C_i}$ are unobservable. Below, we will briefly discuss the natural alternative of choosing the pairs (ρ, ξ) and $(\rho[\xi], \theta)$ as witnesses of these two types of transitions, where ρ is the name of the chosen parametric strand, 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 $\stackrel{\vec{o}}{\longmapsto}{}^*_{\mathcal{S}}$ of $\stackrel{o}{\longmapsto}_{\mathcal{S}}$, where $\vec{o}$ is the sequence of the component moves ("$\cdot$" if empty). $\vec{o}$ 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 $(\sigma_1, \sigma_1^\sharp) \stackrel{o}{\longmapsto}_{\mathcal{S}} (\sigma_2, \sigma_2^\sharp)$, then $(\sigma_2, \sigma_2^\sharp)$ is also a configuration. This property clearly extends to multistep transitions.

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. We conclude this section with an analysis of the notions just defined.

The above definition embeds two distinct notions of traces for strand constructions. On the one hand, the bundle within a configuration gives a precise account of which events have taken place, abstracting from their temporal occurrence order (and instantiation details), but taking into consideration their dependencies both in terms of the ordering of steps (captured by $\Longrightarrow$-edges) and message transmission/reception (expressed by the $\longrightarrow$-arrows). On the other hand, the move sequence $\vec{o}$ that labels the transition arrow also indicates which steps have taken place, but imposes a linear occurrence order on them. We will now relate these two notions.

Notice that each move inserts exactly one node in a configuration. Moreover, the very possibility of making such an insertion is regulated by the two types of edges. Therefore, we can think of a bundle as specifying a partial order of the occurrence of individual moves (the ordering relation is the transitive closure of the union of $\Longrightarrow$ and $\longrightarrow$). Instead, a move sequence linearizes the set of moves into a total order. In general, we can linearize a bundle σ as a sequence of moves in many ways. The following definition imposes constraints on the form of acceptable move sequences.

Given a bundle $\sigma = (S, \Longrightarrow, \longrightarrow)$, we define O_σ as the set of move sequences $\vec{o} = (o_1, \ldots, o_{|S|})$ such that, for $i = 1, \ldots, |S|$, $o_i = (\nu_i, \bar{\nu}_i^p, \bar{\nu}_i^s)$ and

- $\nu_i \in S$ and $\nu_i \neq \nu_j$ for $i \neq j$.
- – if ν_i is initial in S, then $\bar{\nu}_i^p = -$;
  – if there is an index $j < i$ with $o_j = (\nu_j, \bar{\nu}_j^p, \bar{\nu}_j^s)$ such that $\nu_j \Longrightarrow \nu_i$ in σ, then $\bar{\nu}_i^p = \nu_j$.
- – if $\nu_i = +m$, then $\bar{\nu}_i^s = -$;
  – if $\nu_i = -m$ and there is an index $j < i$ with $o_j = (\nu_j, \bar{\nu}_j^p, \bar{\nu}_j^s)$ such that $\nu_j \longrightarrow \nu_i$ in σ, then $\bar{\nu}_i^s = \nu_j$.

Then, any legal move sequence $\vec{o}$ from $(\cdot, \cdot)$ to any configuration containing σ is an element of O_σ. This is formalized in the following completeness result.

Property 3.1 *Let $(\sigma, \sigma^\sharp)$ be a configuration over a set $\mathcal{S}$ of parametric strands and $\vec{o}$ a move sequence such that $(\cdot, \cdot) \stackrel{\vec{o}}{\longmapsto}{}^*_{\mathcal{S}} (\sigma, \sigma^\sharp)$. Then $\vec{o} \in O_\sigma$.*

Proof: We proceed by induction on the length of the move sequence $\vec{o}$, checking that each element in it satisfies the above definition. □

Moreover, any $\vec{o}$ in O_σ is a legal move sequence from $(\cdot, \cdot)$ to any configuration containing σ, as expressed by the following soundness result.

Property 3.2 *Let $(\sigma, \sigma^\sharp)$ be a configuration over a set $\mathcal{S}$ of parametric strands, then for each $\vec{o} \in O_\sigma$, the multistep transition $(\cdot, \cdot) \stackrel{\vec{o}}{\longmapsto}{}^*_{\mathcal{S}} (\sigma, \sigma^\sharp)$ is well-defined.*

Proof: We proceed by induction on the size of the configuration, where $(\sigma_1, \sigma_1^\sharp) \prec (\sigma_2, \sigma_2^\sharp)$ if σ_1 is a proper subgraph of σ_2 or if σ_1 is a subgraph of σ_2 and $\sigma_1^\sharp$ is a proper subgraph of $\sigma_2^\sharp$. □

If $\vec{o}$ describes the transition from $(\cdot, \cdot)$ to a configuration $(\sigma, \sigma^\sharp)$, the individual moves in $\vec{o}$ contain enough information to playback the sequence of moves and exactly reconstruct σ. This is done as follows.

Given a sequence of moves $\vec{o} = (o_1, \ldots, o_{|\vec{o}|})$, with $o_i = (\nu_i, \bar{\nu}_i^p, \bar{\nu}_i^s)$ for $i = 1 \ldots |\vec{o}|$, we define the *strand space associated with* $\vec{o}$, written $\sigma_{\vec{o}}$, as the triple $(S_{\vec{o}}, \Longrightarrow_{\vec{o}}, \longrightarrow_{\vec{o}})$ given as follows:

- $S_{\vec{o}} = \{\nu_i : i = 1 \ldots |\vec{o}|\}$.
- $\Longrightarrow_{\vec{o}} = \{(\nu_i, \nu_j) : \bar{\nu}_j^p = \nu_i\}$.
- $\longrightarrow_{\vec{o}} = \{(\nu_i, \nu_j) : \bar{\nu}_j^s = \nu_i\}$.

Now, if $\vec{o}$ labels a transition from $(\cdot, \cdot)$ to some configuration $(\sigma, \sigma^\sharp)$, then $\sigma_{\vec{o}}$ is isomorphic to σ. We have the following expected result.

Property 3.3 *Let $(\sigma, \sigma^\sharp)$ be a configuration and $\vec{o}$ a move sequence such that $(\cdot, \cdot) \stackrel{\vec{o}}{\longmapsto}{}^*_{\mathcal{S}} (\sigma, \sigma^\sharp)$. Then, $\sigma_{\vec{o}}$ is a bundle and there is a bi-graph isomorphism between $\sigma_{\vec{o}}$ and σ.*

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

The structure of moves we have considered is not sufficient to reconstruct the final configuration of a move sequence. If we are interested in such objects, we need to enrich our definition of move to include the (name of the) parametric strand ρ and the substitutions ξ and θ used when inserting a new strand in a configuration. Simple adaptations of the properties in this section hold in this extended setting. The definitions and proofs become more technical and will be discussed in the full paper [1].

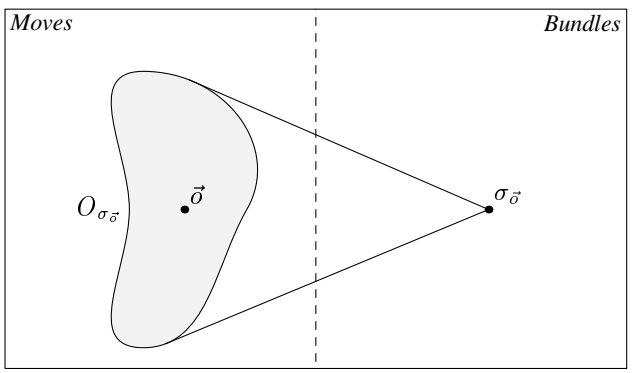


Figure 4. Relating Moves and Bundles

The two constructions we have just defined are essentially inverse of each other, as schematized in Figure 4. Given a bundle, the first returns the set of all the move sequences that produce it. Given a move sequence, the second returns the resulting bundle. In particular, observe that, when starting from a bundle, chaining these transformations yields the same bundle. However, if we start from a move sequence, their cascaded application will return the set of all sequences that construct its same target bundle. These remarks are summarized in the following corollary and Figure 4.

Corollary 3.4 *Let $(\sigma, \sigma^\sharp)$ be a configuration over a set of parametric strands $\mathcal{S}$.*

1. *For every $\vec{o}$ such that $(\cdot, \cdot) \stackrel{\vec{o}}{\longmapsto}{}^*_{\mathcal{S}} (\sigma, \sigma^\sharp)$, we have that $\vec{o} \in O_{\sigma_{\vec{o}}}$.*
2. *For every $\vec{o} \in O_\sigma$, $\sigma_{\vec{o}}$ is isomorphic to σ.*

Proof: The first statement reduces to Property 3.1 after observing that $O_{\sigma_{\vec{o}}} = O_\sigma$ since $\sigma_{\vec{o}}$ is isomorphic to σ. The second part is a consequence of Properties 3.2 and 3.3. □

These considerations allow us to extract a useful notion of equivalence between move sequences: $\vec{o}_1$ and $\vec{o}_2$ are *equivalent* if they produce the same bundle, which can be tested by verifying whether $\sigma_{\vec{o}_1}$ and $\sigma_{\vec{o}_2}$ are isomorphic. The equivalence class to which a move sequence $\vec{o}$ belongs is therefore $O_{\sigma_{\vec{o}}}$. Notice also that, in general, symmetry considerations do not allow selecting a unique element of O_σ as "the" normal move sequence from $(\cdot, \cdot)$ to a configuration embedding a bundle σ: this suggests that $\sigma_{\vec{o}}$ is the most compact representation of the equivalence class $O_{\sigma_{\vec{o}}}$ of $\vec{o}$.

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 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 three transformations which demonstrate that without loss of generality, we can subsequently consider only normalized protocol theories. Note that these transformations are used for mathematical convenience: non-normal 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 $r_{\rho 0} : \pi(\vec{x}) \longrightarrow \mathsf{A}_{\rho 0}(\vec{x}), \pi(\vec{x})$, into the first rule of ρ. For each of its two schematic forms:

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

we obtain the following rules:

- $r'_{\rho 1} : \pi(\vec{x}) \longrightarrow \exists \vec{n}.\, \mathsf{A}_{\rho 1}(\vec{x}, \vec{n}), \mathsf{N}(m(\vec{x}, \vec{n})), \pi(\vec{x})$
- $r'_{\rho 1} : \pi(\vec{x}), \mathsf{N}(m(\vec{x})) \longrightarrow \mathsf{A}_{\rho 1}(\vec{x}), \pi(\vec{x})$

respectively. In both cases, the parameters $\vec{x}$ include the arguments of the elided $\mathsf{A}_{\rho 0}$, and $m(\vec{x})$ does not need to mention each variable in $\vec{x}$. This amounts to setting initial values in the first step of a role, rather than prior to any message exchange.

Nonces: We can 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 the role state 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.

Persistent information: We transform protocol theories so that all persistent information is consulted in the first role, and thus only this first rule mentions persistent predicates. As above, we propagate these data to later rules by adding arguments to the role state predicates. Informally, this amounts to guessing all the persistent information up front, a potentially inefficient approach. This simplifies the formal relationship to strand spaces, but we do not suggest that this transformation be used in practice.

This intuitive description should be detailed enough to spare formalizing this transformation. A fine understanding, especially of the placement of $\exists$, is best gained by interpreting our extended multiset rewriting in linear logic, which is however beyond the scope of this paper. As a result, we are left with the following *normalized rules*:

Role generation rules:

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

Other rules:

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

where we have written the transformed role state predicates with a short line above them. 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 $\mathcal{R}$. Given a state S, we indicate the set of states resulting from this process as T_S: indeed, if S contain the instantiated role predicate $\mathsf{A}_0(a)$ from our specification of the Needham-Schroeder protocol in Figure 1, for each principal B, T_S will contain a state embedding the fact $\bar{\mathsf{A}}_0(a, N_a, B)$, where N_a is the "new" constant introduced by $\exists$.

It is fairly easy to prove that the above transformation is sound and complete with respect to our original definition of a role, even in the presence of the intruder (see Appendix A).

Lemma 4.1 *Let $\mathcal{R}$ be a protocol theory, and S_1 and S_2 two states. Then,*

1. *If $\bar{S}_1 \in T_{S_1}$ and $\bar{S}_2 \in T_{S_2}$, and moreover $\bar{S}_1 \xrightarrow{\vec{r}}{}^*_{\overline{\mathcal{R}}} \bar{S}_2$, then $S_1 \xrightarrow{\vec{r}}{}^*_{\mathcal{R}} S_2$.*
2. *If $S_1 \xrightarrow{\vec{r}}{}^*_{\mathcal{R}} S_2$, then there are $\bar{S}_1 \in T_{S_1}$ and $\bar{S}_2 \in T_{S_2}$ such that $\bar{S}_1 \xrightarrow{\vec{\bar{r}}}{}^*_{\overline{\mathcal{R}}} \bar{S}_2$.*
3. *If $\bar{S}_1 \in T_{S_1}$ and $\bar{S}_2 \in T_{S_2}$, and moreover $\bar{S}_1 \xrightarrow{\vec{\bar{r}}}{}^*_{\mathcal{I}, \overline{\mathcal{R}}} \bar{S}_2$, then $S_1 \xrightarrow{\vec{r}}{}^*_{\mathcal{I}, \mathcal{R}} S_2$.*
4. *If $S_1 \xrightarrow{\vec{r}}{}^*_{\mathcal{I}, \mathcal{R}} S_2$, then there are $\bar{S}_1 \in T_{S_1}$ and $\bar{S}_2 \in T_{S_2}$ such that $\bar{S}_1 \xrightarrow{\vec{\bar{r}}}{}^*_{\mathcal{I}, \overline{\mathcal{R}}} \bar{S}_2$.*

where $\vec{\bar{r}}$ is obtained by normalizing $\vec{r}$.

Proof: By induction on the length of the given transition sequences. □

4.2 Translation

We are now in a position to translate roles expressed in the transition system formalisms into parametric strands. To each normalized role specification $\bar{\rho}$, we associate a parametric strand $\ulcorner \bar{\rho} \urcorner$ of the following form

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

where $\vec{n}$ are the existential variables mentioned in the first rule $\bar{r}_{\rho 1}$ of this role, $\pi(\vec{x})$ are the persistent predicates accessed in this rule, and $\vec{y}$ are the other variables appearing in the role ($\vec{x}, \vec{y}, \vec{n}$ 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):

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

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 3 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. The translation of the intruder model is discussed in Appendix A.

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

- $\mathrm{Fr}(\sigma) = \vec{m}, \vec{m}'$.
- Let $\bar{\mathsf{A}}_{\rho i}(\vec{n_\rho}, \vec{t_\rho})$ in A be the instantiation of the i-th role state predicate of a role $\bar{\rho}$ in $\overline{\mathcal{R}}$ with nonces $\vec{n}_\rho$ and terms $\vec{t_\rho}$. Then,
  - $\sigma^\sharp$ contains a strand $s^\rho(\vec{n_\rho}, \vec{t_\rho})$, obtained by instantiating the strand $s^\rho = \ulcorner \bar{\rho} \urcorner$ with "fresh" constants $\vec{n}_\rho$ and terms $\vec{t_\rho}$.
  - σ contains an initial prefix of $s^\rho(\vec{t})$ 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 (see Appendix A).

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.

Given this definition, we can state the correctness result for our translation of transition systems into strand constructions as follows, where details about how the intruder models are related to each other can be found in Appendix A.

Theorem 4.2 *Let I_0 be some initial intruder knowledge and $\ulcorner I_0 \urcorner$ its strand translation as from Appendix A. If $\Pi, I_0 \xrightarrow{\vec{r}}{}^*_{\mathcal{I},\mathcal{R}} S$ is a multiset rewriting transition sequence over $\mathcal{I}, \mathcal{R}$ from the empty state to state S, then there is a configuration $(\sigma, \sigma^\sharp)$ and a sequence of moves $\vec{o}$ such that*

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

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

Proof: The proof proceeds by induction on $\vec{r}$, mapping every step to a corresponding move in the strand world, while preserving the compatibility relation. □

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 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 $\pi(\vec{x}) \longrightarrow \mathsf{A}_{\rho 0}(\vec{x}), \pi(\vec{x})$, 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 $-\!\!\rightarrow$-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 $\vec{x}_s, \vec{n}_s$ ($\vec{n}_s$ marked *fresh*) with 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 $\vec{x}$ of A_{s0} will be the identity of the principal executing the strand, A say, and the related relevant persistent parameters in $\pi(\vec{x}_s)$ (*e.g.* A's private keys).

Successor arrow to a positive node:

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

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

Successor arrow to a negative node:

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

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

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 5, where the additions have been grayed out, the owner of each strand has been added in its first node, and boxes have been drawn around new persistent information.

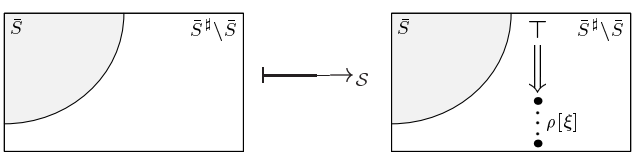
Since we have changed the syntax of a parametric strand, we need to upgrade its dynamics, originally presented in Section 2. First, an obvious alteration to the instantiation of a parametric strand: we apply the substitution to the labels of the $\Longrightarrow$-edges as well as to the messages embedded 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 $\mathsf{A}_{si}(\vec{t})$. We indicate this as $\nu_1 \overset{\mathsf{A}_{si}(\vec{t})}{\Longrightarrow} \nu_2$ (or with its vertical equivalent). Notice that we erased this information in the opposite 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 $\bar{\sigma}_2$

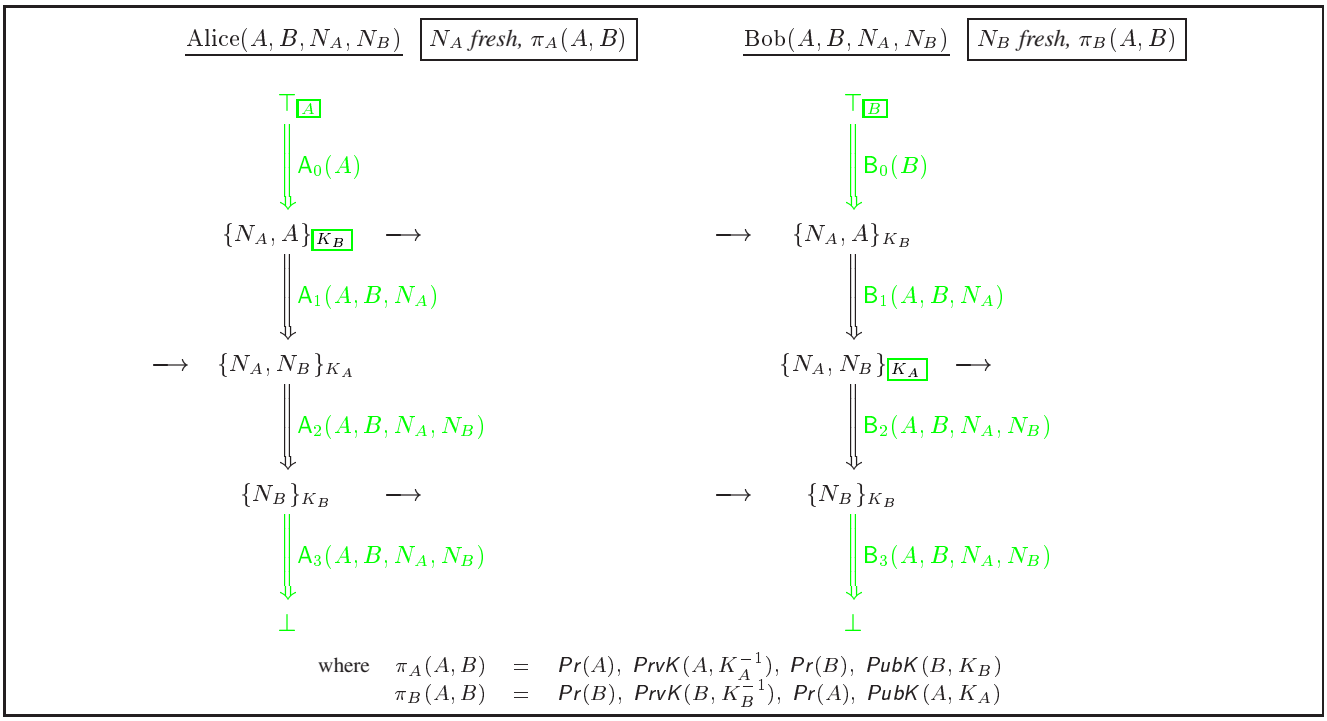
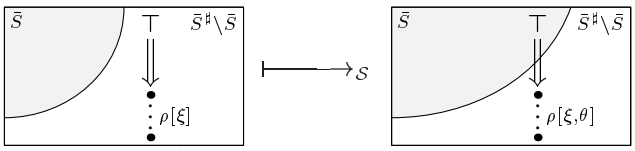


Figure 5. Extended Strand Specification of the Needham-Schroeder Protocol



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 5.1 *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,*

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

where $\vec{\delta}$ is obtained from $\vec{o}$ 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. □

5.2 Translation

We are now in a position to 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 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 6, 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. Applying this translation to the enhanced parametric strands representing the Needham-Schroeder protocol in Figure 5 produces exactly the original transition system specification given in Figure 1.

We will now show that the translation we just outlined preserves transition sequences. In order to do so, 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 Π, A, N, I obtained as follows, where we write $\wr \ldots \int$ for the multiset equivalent of the usual set notation $\{\ldots\}$:

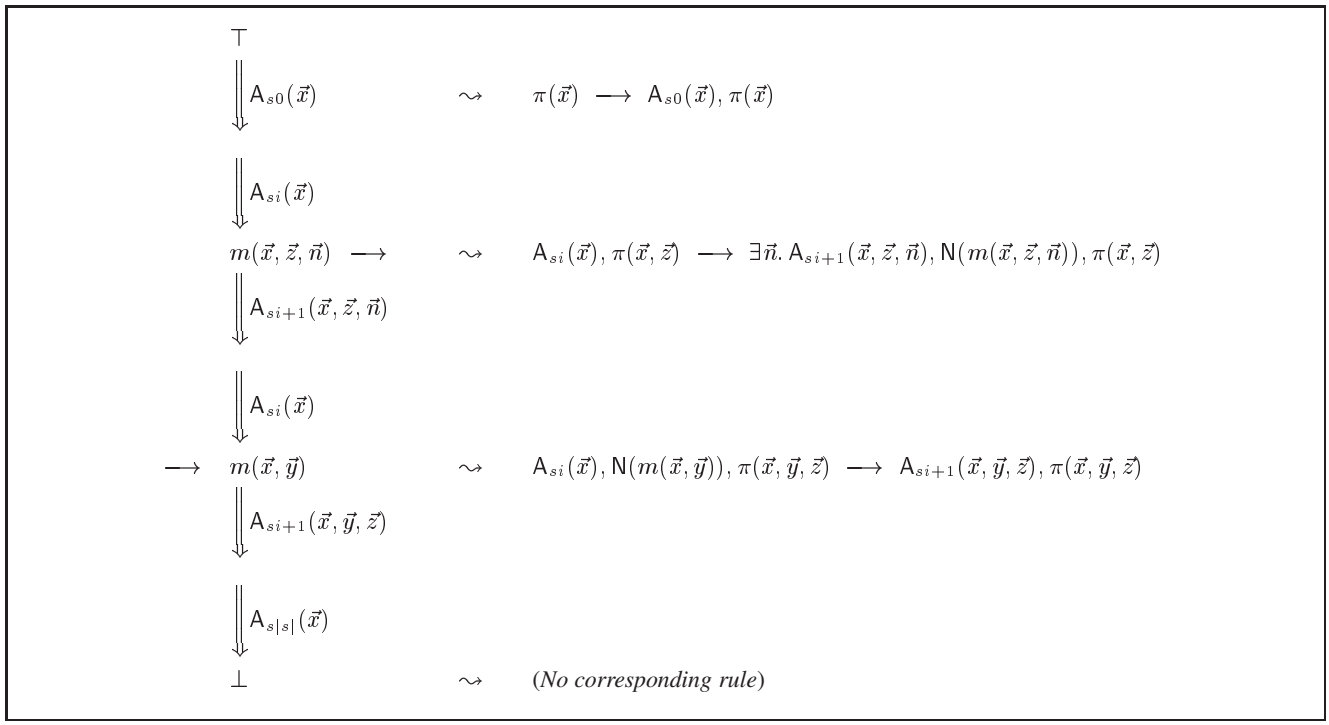


Figure 6. Transforming Extended Strands to Multiset Rewriting Rules

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

Intuitively, we collect the messages in transit coming from honest principal's strands in N, the current knowledge of the intruder in 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 A.

Then, sequences of moves in the strand world and their translation as transition system steps are related as follows, where the treatment of the intruder models is described in Appendix A:

Theorem 5.2 *Let P_0 be some initial penetrator knowledge, and $\ulcorner P_0 \urcorner$ its multiset translation as in Appendix A. 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*

$$(\sigma_1, \sigma_1^\sharp) \stackrel{\vec{\sigma}}{\longmapsto}{}^*_{\mathcal{P}(P_0),\mathcal{S}} (\sigma_2, \sigma_2^\sharp),$$

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

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

Proof: The proof of this result proceeds by induction on the structure of $\vec{o}$. The only non-obvious aspect is that, as observed in Appendix A.4, 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 defer a further discussion of this aspect to the full version of this paper [1].

6 Conclusions and Future Work

We have developed a formal connection between multiset rewriting [2, 5] and strand constructions [6, 10]. 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 immediately 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. 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 can improve the situation further. In fact, we are currently investigating properties of the representation of both strand and multiset rewriting constructions in linear and other nonclassical logics, as well as in process specification languages such as colored Petri nets.

Acknowledgements

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

[1] Iliano Cervesato, Nancy Durgin, Patrick Lincoln, John Mitchell, and Andre Scedrov. A comparison between strand spaces and transition systems for the specification of security protocols. To appear as a technical report, Department of Computer Science, Stanford University.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

A Intruder Models

In this appendix, we describe the intruder models used in the multiset rewriting formalism (Section A.1) and in the strand constructions (Section A.2). In Section A.1, we also propose an alternate formulation of the former, that eases the translation between the two formalisms in Sections A.3 and A.4.

A.1 Intruder Theory

The knowledge available at any instant to the intruder consists of the persistent information in Π, of unused portion of the initial knowledge (*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

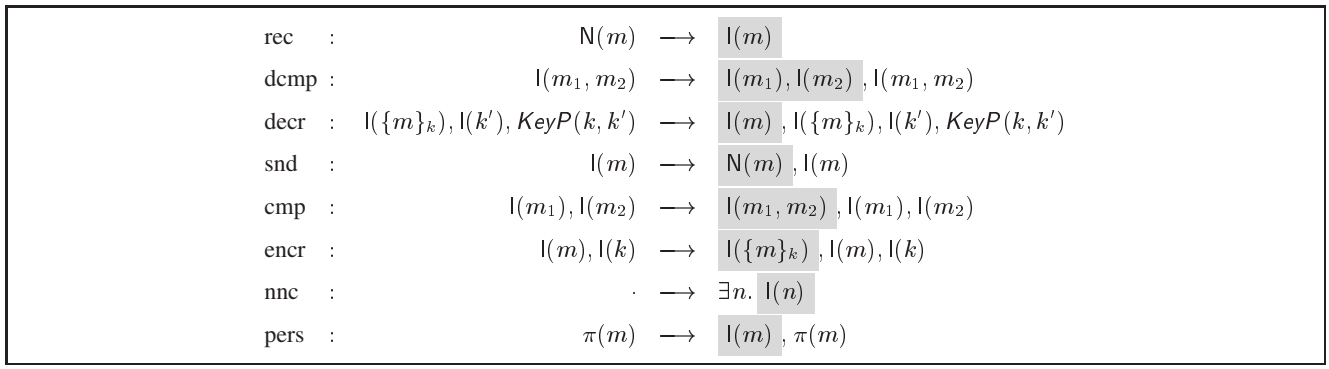


Figure 7. The Standard Intruder Theory $\mathcal{I}$

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 I. We write 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 7. These rule are taken verbatim from [2, 5]. $\mathcal{I}$ implements the Dolev-Yao model [4, 9] 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.

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 rules $\mathcal{I}'$ in Figure 8.

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 A.1 *Let $\mathcal{R}$ be an arbitrary protocol theory, and S_1 and S_2 two states.*

- *If $S_1 \xrightarrow{\vec{r}}{}^*_{\mathcal{R},\mathcal{I}} S_2$, then $S_1 \xrightarrow{\vec{r}'}{}^*_{\mathcal{R},\mathcal{I}'} S_2$.*
- *If $S_1 \xrightarrow{\vec{r}'}{}^*_{\mathcal{R},\mathcal{I}'} S_2$, then there is an intruder state I' such that $S_1 \xrightarrow{\vec{r}}{}^*_{\mathcal{R},\mathcal{I}} S_2, I'$.*

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 $\vec{r}'$ is derived from $\vec{r}$ according to this strategy.

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 I' consists of copies of intermediate information produced by the rules of $\mathcal{I}$ plus whatever data were explicitly discarded using del. □

A.2 Penetrator Strands

We now formalize the intruder model of [6, 10], 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 9, which includes a case to handle intruder-generated nonces. This possibility is missing from [6, 10], but the completion is straightforward. We also distinguished cases $M(m)$ and $M'(m)$, which are identified in [6, 10]. We refer to the collection of (parametric) penetrator strands in Figure 9 as $\mathcal{P}(P_0)$.

Several observations need to be made. First, the intruder specification underlying penetrator strands follows the Dolev-Yao model [4, 9]. The parametric strands in Figure 9 are indeed closely related to the intruder model $\mathcal{I}'$ above. A translation can be found in Sections A.3 and A.4 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

$$
\begin{array}{lrcl}
\mathrm{rec}' & : & \mathsf{N}(m) & \longrightarrow & \mathsf{I}(m) \\
\mathrm{dcmp}' & : & \mathsf{I}(m_1, m_2) & \longrightarrow & \mathsf{I}(m_1), \mathsf{I}(m_2) \\
\mathrm{decr}' & : & \mathsf{I}(\{m\}_k), \mathsf{I}(k'), \mathsf{KeyP}(k, k') & \longrightarrow & \mathsf{I}(m), \mathsf{KeyP}(k, k') \\
\mathrm{snd}' & : & \mathsf{I}(m) & \longrightarrow & \mathsf{N}(m) \\
\mathrm{cmp}' & : & \mathsf{I}(m_1), \mathsf{I}(m_2) & \longrightarrow & \mathsf{I}(m_1, m_2) \\
\mathrm{encr}' & : & \mathsf{I}(m), \mathsf{I}(k) & \longrightarrow & \mathsf{I}(\{m\}_k) \\
\mathrm{nnc}' & : & \cdot & \longrightarrow & \exists n.\, \mathsf{I}(n) \\
\mathrm{pers}' & : & \pi(m) & \longrightarrow & \mathsf{I}(m), \pi(m) \\
\mathrm{dup} & : & \mathsf{I}(m) & \longrightarrow & \mathsf{I}(m), \mathsf{I}(m) \\
\mathrm{del} & : & \mathsf{I}(m) & \longrightarrow & \cdot
\end{array}
$$

Figure 8. The Modified Intruder Theory $\mathcal{I}'$

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.

A.3 From Intruder Theory to Penetrator Strands

The introduction of the alternate intruder theory $\mathcal{I}'$ in Section A.1 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:

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

where we have equipped the intruder rules with arguments in the obvious way. We also need to map I_0 to a set P_0 of messages initially known to the intruder, to be processed by the penetrator strand M': $\ulcorner I_0 \urcorner = \{m : \mathsf{I}(m) \in I_0\}$. Every access to a message $\mathsf{I}(m)$ in I_0 will be translated to an application of the penetrator strand $M'(m)$.

A.4 From Penetrator Strands to Intruder Theory

The translation of the penetrator strands $\mathcal{P}(P_0)$ in Figure 9 essentially is the inverse of the above mapping. Our target intruder model, in the multiset rewriting world, is $\mathcal{I}'$.

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

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 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.

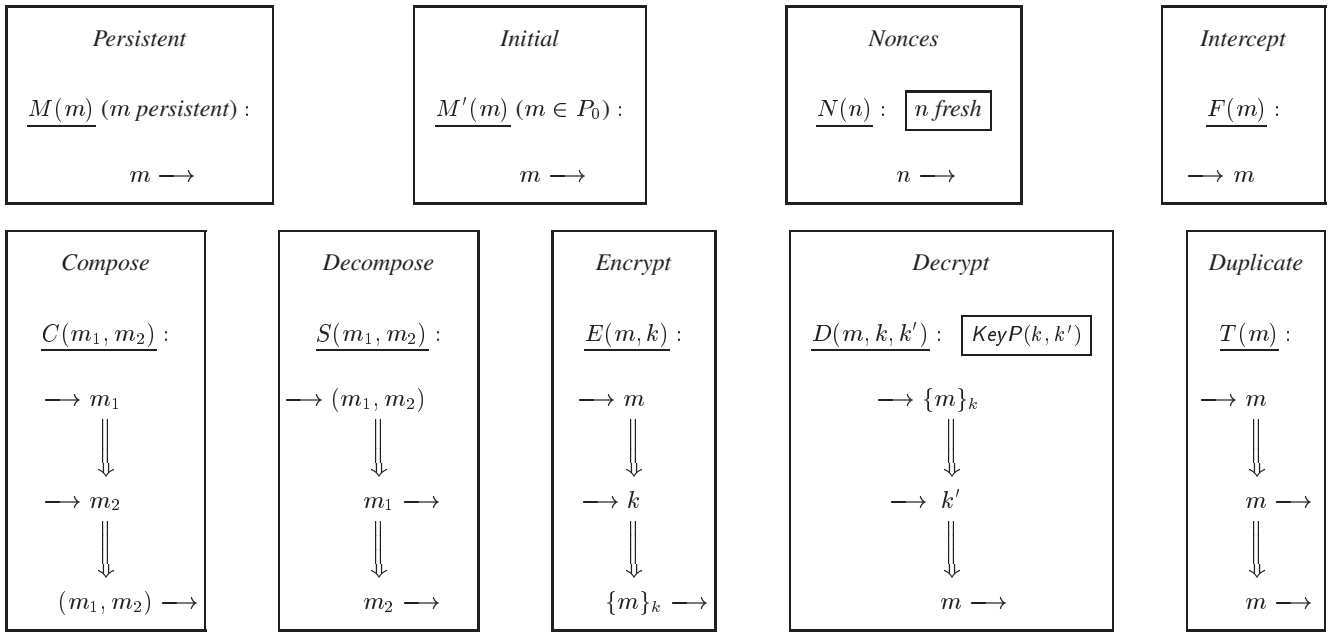


Figure 9. The Penetrator Strands $\mathcal{P}$