On the Relationship between Strand Spaces and Multi-Agent Systems

JOSEPH Y. HALPERN and RICCARDO PUCELLA
Cornell University

Strand spaces are a popular framework for the analysis of security protocols. Strand spaces have some similarities to a formalism used successfully to model protocols for distributed systems, namely *multi-agent systems*. We explore the exact relationship between these two frameworks here. It turns out that a key difference is the handling of agents, which are unspecified in strand spaces and explicit in multi-agent systems. We provide a family of translations from strand spaces to multi-agent systems parameterized by the choice of agents in the strand space. We also show that not every multi-agent system of interest can be expressed as a strand space. This reveals a lack of expressiveness in the strand-space framework that can be characterized by our translation. To highlight this lack of expressiveness, we show one simple way in which strand spaces can be extended to model more systems.

Categories and Subject Descriptors: D.4.6 [**Operating Systems**]: Security and Protection; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; K.4.4 [**Computers and Society**]: Electronic Commerce—*Security*

General Terms: Security, Theory

Additional Key Words and Phrases: Agents, expressiveness, multi-agent systems, security protocols, strand spaces,

1. INTRODUCTION

Strand spaces [Thayer et al. 1999b] (THG from now on) have recently emerged as a popular framework for the analysis of security protocols. Roughly speaking, the strand space corresponding to a protocol is the set of the traces of the various interactions between the principals under consideration. Using strand spaces, we can reason about the secrecy of the values exchanged between principals and infer authentication properties. One limitation of the strand-space approach is that it assumes that all the information available to a principal is either supplied initially or contained in messages received by that principal. However, there is other important

A preliminary version of this paper appears in the *Proceedings of the 8th ACM Conference on Computer and Communications Security*, 2001. Supported in part by NSF under grant IRI-96-25901 and IIS-0090145 and by ONR under grants N00014-00-1-03-41 and N00014-01-10-511, and by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the ONR under grant N00014-01-1-0795. Authors' address: J. Y. Halpern, Department of Computer Science, Cornell University, Ithaca, NY 14853, email: `halpern@cs.cornell.edu`, home page: `http://www.cs.cornell.edu/home/halpern`, R. Pucella, Department of Computer Science, Cornell University, Ithaca, NY 14853, email: `riccardo@cs.cornell.edu`.

information that may also be available in a security setting. For example, an adversary may have information about the protocol(s) being used. Moreover, if the same agent is playing different roles, then it may be able to combine information it gathers in its various roles. This information can be captured precisely using a formal model of knowledge. Indeed, the *multi-agent systems* framework used to represent the knowledge and belief of agents has been used quite successfully to reason about distributed protocols (see [Fagin et al. 1995] for intuition, details, and examples). This framework is based on a notion of *runs*; a run is a complete description of what happens over time in one possible execution of the system. Early attempts at applying the multi-agent systems framework to reasoning about cryptography and security (cf. [Gray III and Syverson 1998; Halpern et al. 1988]) suggest that these notions of knowledge and belief can be an important component in reasoning about security, the BAN logic being an example in that particular direction [Burrows et al. 1990]. Essentially, the idea is simply that information can be derived in protocols not just through the messages being exchanged, but through general properties of the system. Our current project is to define a multi-agent systems framework suitable for reasoning about security using notions such as knowledge and belief. On the other hand, strand spaces have been used successfully to reason about security protocols. Since there are similarities between the two approaches, it is worthwhile to see how much of the strand-space approach can be carried over to multi-agent systems and vice versa. This forces us to investigate in detail the relationship between the two approaches. That is the purpose of this paper.

The key issue in relating the two frameworks is the handling of agents. For our purposes, an agent is an entity (a principal, a process, etc.) that can participate in interactions. This notion of agent is general enough to capture different intuitions, depending on the kind of system being analyzed. Typically, an agent corresponds to a system-independent entity such as a principal on behalf of whom interactions are performed. For our purposes, what matters is that an agent has a state that is shared across all the interactions that the agent performs. In multi-agent systems, there is a clear notion of an agent participating in an interaction. In strand spaces, there is not. Each protocol interaction (described by a strand) is viewed as independent from all others. In fact, each strand can be viewed as representing a different agent. This approach to modeling agents is deliberate in the definition of strand spaces, and gives a theory that yields general results. Strand spaces do treat agents, in a fashion, by essentially assigning to every strand a name representing the "agent" executing the strand; see, for instance, the description of NSL spaces by THG used to model the Needham-Shroeder-Lowe protocol. However, it is still the case that strands corresponding to the same "agent" can exchange values only through explicit communication, i.e. there is no shared state across the strands corresponding to the same "agent" name. For all intents and purposes, these strands may as well be assigned to different actual agents.

To highlight the role of agents, we provide a family of translations from strand spaces to *strand systems*, a subclass of multi-agent systems that seem to capture the intuition underlying strand spaces. The translations are parameterized by an assignment from strands to agents. This assignment associates with a strand the agent performing the protocol interaction described by the strand. Such an assignment captures the intuition that different strands can potentially be executed by the same agent.

Why is the role of the agents so significant? For the protocols considered by THG, it is not. On the other hand, it is clear from the work on BAN [Burrows et al. 1990] and other logics (for instance, [Stubblebine and Wright 1996; Syverson 1990]), as well as the work on information flow [McLean 1994], that belief and knowledge are useful concepts when reasoning about security protocols. As we said earlier, there are a number of ways that an attacker can gain knowledge in a system. Certainly when an attacker intercepts a message, it learns the contents of the message. But it may learn much more if it knows the protocol being run. In addition, different principals representing the same attacker may be able to pool the information they have acquired. In any case, as soon as one talks about belief or knowledge, there must be agents in the picture to which belief or knowledge is ascribed. One advantage of a multi-agent system is that it explicitly identifies agents and provides an easy way to ascribe knowledge to agents (see [Fagin et al. 1995]). In the context of security, that means we are forced to reason about, for example, which principals represent the same agent or which ones may represent the same agent. (See [Grove 1995; Grove and Halpern 1993] for logics that carry out such reasoning explicitly and, in particular, distinguish between agents and their names.)

Significantly, our translations are not surjective. Some strand systems are not the image of any strand space, regardless of the assignment of agents to strands. This is not just an artifact of our particular translation. Any translation from strand spaces to strand systems that preserves the message history of the agents, in a precise sense, cannot be surjective. Intuitively, this is because in a strand space we cannot say "either this sequence of events happens or that one does, but not both". This indicates a fundamental lack of expressiveness in the current formulation of strand spaces.

One way to characterize this lack of expressiveness is by showing how strand spaces can be extended to be able to model arbitrary strand systems. We demonstrate one way of doing this by introducing a notion of *conflict*, specifying when two strands cannot both be part of the same run. We remark that the general properties of strand spaces proved by THG, such as the bounds on the penetrator, are still valid in these extended strand spaces. We believe that this notion of conflict becomes important when considering modern security protocols. Protocols such as SSL or TLS involve the selection of a subprotocol during the execution of a protocol instance. Since only one such subprotocol can be chosen, it is natural to use conflict to model this.

Despite this lack in expressiveness, strand spaces are quite successful at analyzing typical protocols, particularly authentication protocols. Intuitively, based on the discussion above, this should be due to those protocols not making any choice. We formalize this intuition by exhibiting a property of protocols that ensures that a strand system generated from a protocol with such a property (using established techniques) is in fact the image of a strand space under the natural translation.

The rest of this paper is structured as follows. In Section 2, we review strand spaces and multi-agent systems. In Section 3, we present the translation from strand spaces to strand systems. In Section 4, we discuss the problem of translating a strand system into a strand space, and show why in general we cannot perform the translation faithfully. In Section 5, we describe an extension to the strand space framework that is equivalent in expressive power to strand systems. In Section 6, we discuss the generation of systems from protocols. We interpret our results in Section 7. The proof of all technical results can be found in the Appendix.

2. THE FRAMEWORKS

In this section, we review the two frameworks we want to relate, the strand-space framework of THG, and the multi-agent systems framework [Fagin et al. 1995].

2.1 Strand spaces

Let M be the set of possible messages that can be exchanged by the principals in a protocol.[1] A *signed term* is a pair $\langle \sigma, u \rangle$ with $\sigma \in \{+, -\}$ and $u \in M$. A signed term $\langle +, u \rangle$ represents the sending of message u and is typically written $+u$, and a signed term $\langle -, u \rangle$ represents the reception of message u and is typically written $-u$. We write $(\pm M)^*$ for the set of finite sequences of signed terms. A *strand space* over M consists of a set Σ, whose elements are called *strands*, together with a trace mapping $\mathrm{tr} : \Sigma \rightarrow (\pm M)^*$, associating each strand in Σ with a sequence of signed terms. We typically represent a strand space by the underlying set Σ, leaving the trace mapping implicit.

In a strand space Σ, a *node* is a pair $\langle s, i \rangle$, with $s \in \Sigma$ and an integer i with $1 \leq i \leq |tr(s)|$. The set of nodes of Σ is represented by $\mathcal{N}$. We say the node $\langle s, i \rangle$ *belongs to* the strand s, and sometimes abuse notation by writing $\langle s, i \rangle \in s$. Given a node $n = \langle s, i \rangle$, where $\mathrm{tr}(s) = \langle \sigma_1, u_1 \rangle \ldots \langle \sigma_k, u_k \rangle$, define $\mathrm{term}(n) = \langle \sigma_i, u_i \rangle$. If n_1 and n_2 are nodes, we write $n_1 \rightarrow n_2$ if $\mathrm{term}(n_1) = +u$ and $\mathrm{term}(n_2) = -u$; we write $n_1 \Rightarrow n_2$ if both n_1 and n_2 occur on the same strand s and $n_1 = \langle s, i \rangle$ and $n_2 = \langle s, i+1 \rangle$. Note that the set $\mathcal{N}$ of nodes together with both sets of edges $n_1 \rightarrow n_2$ and $n_1 \Rightarrow n_2$ forms a directed graph $(\mathcal{N}, (\rightarrow \cup \Rightarrow))$.

Strand spaces are aimed at reasoning about the security of systems in the presence of a hostile penetrator with various capabilities. In order to model such a penetrator, a notion of an *infiltrated strand space* is defined by THG; the infiltrated strand space contains both regular strands and a set of so-called *penetrator strands* that represent the actions available to a penetrator. For the purposes of this paper, there is no need to distinguish penetrator strands from regular strands, so we do not consider infiltrated strand spaces.

A bundle represents a snapshot of a possible protocol execution. For a given strand space Σ, let $B = (\mathcal{N}_B, (\rightarrow_B \cup \Rightarrow_B))$ be a subgraph of $(\mathcal{N}, (\rightarrow \cup \Rightarrow))$. The graph B is a *bundle* if

B1. B is finite,

B2. if $n_2 \in \mathcal{N}_B$ and $\mathrm{term}(n_2)$ is negative, then there is a unique n_1 such that $n_1 \rightarrow_B n_2$,

[1] The actual contents of the message and the structure of M are not important for the purpose of this paper.

B3. if $n_2 \in \mathcal{N}_B$ and $n_1 \Rightarrow n_2$, then $n_1 \Rightarrow_B n_2$,

B4. B is acyclic.

In B2 and B3, because B is a graph, it follows that $n_1 \in \mathcal{N}_B$. We say a node n is in the bundle B if it is in $\mathcal{N}_B$.

It will be useful for us in this paper to allow infinite bundles. An *infinite bundle* is just a subgraph of $(\mathcal{N}, (\rightarrow \cup \Rightarrow))$ that satisfies B2–4 (that is, we no longer require the finiteness condition B1). The *height* of an infinite bundle is the length of the longest finite sequence of nodes $n_1, n_2, n_3, \ldots, n_k$ in B such that $n_1 \rightsquigarrow n_2 \rightsquigarrow \ldots \rightsquigarrow n_k$, where $\rightsquigarrow$ is either $\rightarrow$ or $\Rightarrow$. (A bundle can have infinite height if there is no bound on the length of the longest sequence of this type.) Of course, all finite bundles have finite height. It is easy, however, to construct infinite bundles of infinite height (even if all individual strands have length at most 2). For example, consider the strand space $\Sigma = \{s_i : i \in \mathbb{Z}\}$, with a trace mapping $\mathrm{tr}(s_i) = \langle -u_i, +u_{i+1} \rangle$. The strand space Σ itself in this case is an infinite bundle of infinite height. All the arguments of THG which were applied to finite bundles go through without change for infinite bundles of finite height. (Indeed, they go through for infinite bundles that are *well-founded*, in the sense of having no infinite "descending" sequences of the form $\ldots \rightsquigarrow n_3 \rightsquigarrow n_2 \rightsquigarrow n_1$, although we end up using only bundles of finite height in our arguments.)

2.2 Multi-agent systems

In the multi-agent systems approach, every agent is assumed to be in some *local state* at each point in time. Given a set $\mathcal{A}$ of agents, we characterize a system over $\mathcal{A}$ at a given point in time in terms of a *global state*; this is a tuple $\langle \sigma_a : a \in \mathcal{A} \rangle$, where σ_a is the local state of agent a. The local states of an agent intuitively encode all the information that the agent has available at a given point in time. In typical distributed systems applications, the local state includes the values of variables and a history of messages received. If we are modeling a group of agents playing a poker game, the local state may include the cards that the agent holds and the bets that have been made thus far.

To capture changes to the system over time, we define a *run* of the system to be a function from time to global states. Intuitively, a run is a complete description of what happens over time in one possible execution of the system. A *point* is a pair (r, m) consisting of a run r and a time m. The global state $r(m)$ describes the state of the system at the point (r, m). Formally, we take a *system* to consist of a set of runs. Informally, the system includes all the possible executions of the system, that is, all the different ways it could evolve through time.

Due to the assumptions made by the strand-space approach, namely that events in strands consist of sending and receiving messages, we consider only systems where the local state of an agent is the sequences of messages that the agent has sent and received. Thus, we deliberately ignore internal actions (or, more accurately, treat them as irrelevant).

We can formalize the above description as follows. Consider a fixed set M of messages. A *history* for agent a (over M) is a sequence of elements of the form $\mathsf{sent}(u)$ and $\mathsf{recv}(u)$, where $u \in M$. We think of $\mathsf{sent}(u)$ as representing the event "message u is sent" and $\mathsf{recv}(u)$ as representing the event "message u is received."

Intuitively, a's history at (r, m) consists of a's initial state, which we take to be the empty sequence, followed by the sequence describing a's actions up to time m. If a performs no actions in round m, then its history at (r, m) is the same as its history at $(r, m-1)$.[2] In such a *message-passing system*, we speak of $\mathsf{sent}(u)$ and $\mathsf{recv}(u)$ as *events*. For $a \in \mathcal{A}$, let $r_a(m)$ be agent a's history in (r, m). We say that an event e *occurs in a's history in round $m+1$ of run r* if e is in (the sequence) $r_a(m+1)$ but not in $r_a(m)$.

In a message-passing system, the agent's local state at any point is its history. Of course, if h is the history of agent a at the point (r, m), then we want it to be the case that h describes what happened in r up to time m from a's point of view. To do this, we need to impose some consistency conditions on global states. In particular, we want to ensure that message histories do not shrink over time, and that every message received in round m corresponds to a message that was sent at some earlier round.

Given a set M of messages, we define a *message-passing system* (over M) to be a system such that for each point (r, m) and each agent $a \in \mathcal{A}$, the following constraints are satisfied:

MP1. $r_a(m)$ is a history over M;

MP2. for every event $\mathsf{recv}(u)$ in $r_a(m)$ there exists a corresponding event $\mathsf{sent}(u)$ in $r_b(m)$, for some $b \in \mathcal{A}$;[3]

MP3. $r_a(0)$ is the empty sequence and $r_a(m+1)$ is either identical to $r_a(m)$ or the result of appending one event to $r_a(m)$.

MP1 says that an agent's local state is its history, MP2 guarantees that every message received at round m corresponds to one that was sent earlier, and MP3 guarantees that histories do not shrink.

We think of strand spaces as completely asynchronous message-passing systems. Roughly speaking, strand spaces do not place any constraints on the relative order of events in different agents' histories beyond those imposed by MP1 and MP2. As argued in [Fagin et al. 1995, Section 4.4.6], we can capture such asynchrony by considering systems that consist of *all* runs satisfying MP1–3 for some set of histories. Formally, we say that $\mathcal{R}$ is a *strand system* if there exists a sequence $\langle V_a : a \in \mathcal{A}\rangle$, where V_a is a set of histories over some set M of messages, such that $\mathcal{R}$ consists of all runs satisfying MP1–3 where agent a's local state is a history in V_a at every point. We call $\mathcal{R}$ the strand system *generated by* $\langle V_a : a \in \mathcal{A}\rangle$. Informally, the set V_a specifies the possible histories agent a could have. The strand system generated by $\langle V_a : a \in \mathcal{A}\rangle$ consists of all runs satisfying MP1–3 such that agent a's histories are in V_a for all $a \in \mathcal{A}$.

Strand systems are closely related to the *asynchronous message-passing systems (amps)* defined in [Fagin et al. 1995, Chapter 4]. The main difference is that for strand systems, messages are anonymous. A message does not specify a sender or a receiver. Messages in amps, on the other hand, are not anonymous. Events have

[2]Round m takes place between time $m-1$ and time m. Actions are performed during a round. The effect of an action performed by agent a at round m appears in agent a's state at time m.
[3]To simplify our translations, we allow an agent to send a message to itself, so a and b can be the same agent.

the form $\mathsf{sent}(u, a, b)$ (u is sent to a by b) and $\mathsf{recv}(u, a, b)$ (u is received by b from a). The remaining differences are minor. Strand systems allow for an infinite number of agents, whereas in amps there are only finitely many agents. Amps can be easily modified so as to allow infinitely many agents. Moreover, agents are allowed in amps to have a nontrivial initial state, while for strand systems, the initial state is always the empty sequence. This was done for compatibility with the definitions of THG.

3. TRANSLATING STRAND SPACES TO STRAND SYSTEMS

In this section, we consider the problem of translating strand spaces into strand systems. We do this by formalizing the strand space intuition that bundles represent snapshots of possible executions. Our construction derives the possible execution traces in terms of sequences of bundles, which are then used to construct the runs of the system.

A multi-agent system requires an explicit set of agents; a strand space does not. To perform the translation, we specify a set $\mathcal{A}$ of agents and a particular *agent assignment* $A : \Sigma \rightarrow \mathcal{A}$, which intuitively associates with each strand $s \in \Sigma$ the agent $A(s)$ executing s. In the generated strand system, an agent behaves as if it were concurrently executing the various strands assigned to it. The motivation behind this approach is that if the same agent is in reality executing many strands, then it should share its knowledge across all the strands it is executing.

The choice of agents and the agent assignment for a given strand space is left to the model designer. Different choices lead to different multi-agent systems. As we show at the end of this section, associating a different agent with each strand enforces the basic strand space tenet that information is exchanged only through explicit messages, i.e. there is no shared state between different strands.

The translation takes as arguments a strand space Σ, a set $\mathcal{A}$ of agents, and an agent assignment A from strands in Σ to agents. To define the translation, we first define a relation on bundles that represents the actions that the agents in the strand space can perform. Given a strand $s \in \Sigma$ and a bundle B, let B-height(s) be the largest i such that $\langle s, i\rangle \in \mathcal{N}_B$. (We take B-height$(s) = 0$ if no node in s appears in B.)[4] A function $f : \Sigma \rightarrow \Sigma$ *respects* A if $A(s) = A(f(s))$, that is, the same agent is associated with both strands s and $f(s)$ for all strands $s \in \Sigma$. If B_1, B_2 are (possibly infinite) bundles of Σ, and $f : \Sigma \rightarrow \Sigma$ is a bijection that respects A, we write $B_1 \sqsubseteq_f B_2$ if the following two conditions hold:

(1) if $\langle s, i\rangle$ is in B_1, then $\langle f(s), i\rangle$ is in B_2 and $\mathrm{term}(\langle s, i\rangle) = \mathrm{term}(\langle f(s), i\rangle)$,

(2) if $\langle s, i\rangle \rightarrow \langle s', j\rangle$ is an edge in B_1, then $\langle f(s), i\rangle \rightarrow \langle f(s'), j\rangle$ is an edge in B_2.

These clauses guarantee that the prefix of s that is in B_1 is a prefix of the prefix of $f(s)$ that is in B_2. For example, if B_1 consists of the single node $\langle s, 1\rangle$ and B_2 consists of $\langle s', 1\rangle$ and $\langle s', 2\rangle$, where $\mathrm{term}(\langle s, 1\rangle) = \mathrm{term}(\langle s', 1\rangle)$, then $B_1 \sqsubseteq_f B_2$, where f is the bijection that permutes s and s', while acting as the identity on all other strands.

[4]This notion of height of a strand in a bundle should not be confused with the notion of height of a bundle we defined in the previous section.

For many cases of interest, we can simply take the bijection f to be the identity; in that case, $B_1 \sqsubseteq_f B_2$ if and only if B_1 is a subgraph of B_2. We discuss the reason for allowing arbitrary bijections and the role of the bijection at the end of this section.

We write $B_1 \mapsto B_2$ if there is a bijection $f : \Sigma \to \Sigma$ that respects A such that

(1) $B_1 \sqsubseteq_f B_2$, and

(2) $\sum_{s \in A^{-1}(a)} B_2\text{-height}(f(s)) - B_1\text{-height}(s) \leq 1$ for all agents $a \in \mathcal{A}$.

Informally, $B_1 \mapsto B_2$ if, for each agent $a \in \mathcal{A}$, B_2 extends the prefix of at most one strand in B_1 corresponding to a, and extends it by at most one node. (Note that the strand $f(s)$ in B_2 extending the prefix of strand s in B_1 may be different from s, depending on the definition of f.) If B_2 does extend the prefix of one of the strands in B_1 corresponding to agent a by one node, let $e_{a,B_1 \mapsto B_2}$ denote the event corresponding to that node: if the node is n and $\text{term}(n) = +u$, then $e_{a,B_1 \mapsto B_2}$ is $\mathsf{sent}(u)$, and if $\text{term}(n) = -u$, then $e_{a,B_1 \mapsto B_2}$ is $\mathsf{recv}(u)$. We define a $\mapsto$-*chain* (or simply a chain) to be an infinite sequence of bundles $B_0, B_1, \ldots$ such that B_0 is the empty bundle and $B_0 \mapsto B_1 \mapsto \ldots$.

Let $\mathit{Chains}(\Sigma, \mathcal{A}, A)$ be the set of all chains in Σ. We associate with every chain in $\mathit{Chains}(\Sigma, \mathcal{A}, A)$ a run as follows: Given a chain $C = B_0 \mapsto B_1 \mapsto \ldots$ and an agent $a \in \mathcal{A}$, define $\mathit{hist}_a^m(C)$ inductively. Let $\mathit{hist}_a^0(C) = \langle\,\rangle$; let $\mathit{hist}_a^{n+1}(C) = \mathit{hist}_a^n(C)$ if no strand corresponding to agent a in B_n is extended in B_{n+1}; otherwise, let $\mathit{hist}_a^{n+1}(C) = \mathit{hist}_a^n(C) \cdot e_{a,B_n \mapsto B_{n+1}}$. (Informally, $\mathit{hist}_a^{n+1}(C)$ is the result of appending to $\mathit{hist}_a^n(C)$ the unique event performed by agent a in going from B_n to B_{n+1}.) Thus, $\mathit{hist}_a^n(C)$ consists of all the events that a has performed in B_n. Let r^C be the run such that $r_a^C(m) = \mathit{hist}_a^m(C)$ and let $\mathcal{R}(\Sigma, \mathcal{A}, A) = \{r^C : C \in \mathit{Chains}(\Sigma, \mathcal{A}, A)\}$.

Theorem 3.1. *$\mathcal{R}(\Sigma, \mathcal{A}, A)$ is a strand system.*

In light of Theorem 3.1, define the map T_A from strand spaces to strand systems by taking $T_A(\Sigma) = \mathcal{R}(\Sigma, \mathcal{A}, A)$.

As we mentioned at the beginning of this section, we can model strand spaces as discussed by THG by taking the set of agents of a strand space Σ to be Σ, and taking the identity function id as the agent assignment. This captures explicitly the intuition that strands are independent protocol executions, that for all intents and purposes may be assumed to be executed by different agents. This is the case since there is no state shared between strands, and every communication is made explicit. In other words, there is no conceptual difference between two strands s_1 and s_2 executed by different processes of an agent or by two distinct agents if there cannot be any shared state between s_1 and s_2.

There is a small amount of information that is lost in the translation from strand spaces to strand systems, which will become evident in Theorem 3.2 below. This loss stems from the fact that messages in strand systems are completely anonymous. For example, if agent 2 and agent 3 both send a message u and later agent 1 receives it, there is no way in a strand system to tell if agent 1 received u from agent 2 or agent 3. By way of contrast, in a strand space, there is an edge indicating who agent 1 received the message from. The multi-agent system framework can in fact keep track of who an agent received a message from by adding an additional component to the global state; this is the state of the *environment*, which intuitively describes everything relevant to the system not included in the local states of the processes.[5] We will not bother going into the details of the environment in this paper, as the issue does not affect our results. We can characterize the information loss resulting from our translation by defining a relation between globals states of $\mathcal{R}(\Sigma, \Sigma, id)$ and bundles of Σ. We say that a global state $\langle \sigma_s : s \in \Sigma \rangle$ (recall that here $\mathcal{A} = \Sigma$) is *message-equivalent* to a bundle B if for each $s \in \Sigma$, if $\sigma_s = \langle e_1, \ldots, e_k \rangle$ then B-height$(s) = k$ and, for each i such that $1 \leq i \leq k$, if $\mathrm{term}(\langle s, i \rangle) = +u$ then e_i is $\mathsf{sent}(u)$, and if $\mathrm{term}(\langle s, i \rangle) = -u$ then e_i is $\mathsf{recv}(u)$. Intuitively, a global state is message-equivalent to any bundle that has the same nodes. This captures the intuition that an agent receiving a message is not aware of the sender. The following theorem shows that, except for this loss of information, our translation from strand spaces to strand systems essentially identifies bundles and global states (if we treat all strands as being associated with a different agent).

THEOREM 3.2. *Every global state of $\mathcal{R}(\Sigma, \Sigma, id)$ is message-equivalent to a bundle of Σ of finite height, and every bundle of Σ of finite height is message-equivalent to a global state of $\mathcal{R}(\Sigma, \Sigma, id)$.*

We remark that if the environment state is used to record the sender of each received message, Theorem 3.2 can be strengthened to a 1-1 correspondence between global states of $\mathcal{R}(\Sigma, \Sigma, id)$ and bundles of Σ of finite height.

With these results in hand, we now discuss some of the choices made, in particular, why we allowed infinitely many agents, infinite bundles, and an arbitrary bijection f in the definition of $\mapsto$. It turns out that these choices are somewhat related. First observe that, in Theorem 3.2, we identified each strand with an agent. Thus, if there are infinitely many strands in the strand space, the corresponding strand system requires infinitely many agents. Naturally, if we restrict our analysis to strand spaces with only finitely many strands, then we can take the corresponding strand systems to have only finitely many agents. Infinite bundles are needed in order to prove Theorem 3.1 when there are infinitely many agents. To understand why, consider a strand space Σ, where $\Sigma = \{s_1, s_2, \ldots\}$ and $\mathrm{tr}(s_n) = \langle +u_n \rangle$. In other words, strand s_n has exactly one node, at which a send action is performed. If a different agent is associated with each strand, then in the corresponding strand system, the set of histories for agent n will consist of the empty history and the history $\langle \mathsf{sent}(u_n) \rangle$. The system based on this set of histories has a run where all the agents send their message simultaneously at round 1. This history corresponds to the infinite bundle consisting of all the strands in Σ. Intuitively, if all the agents can send a message, there is no reason that they should not all send it the first round.

Why do strand spaces allow infinitely many strands? Often, security protocols rely on *nonce values*, which are values guaranteed to be unique within a run of the system. Strand spaces model nonce values by specifying a different strand for each possible value of a nonce. Since, theoretically, there can be infinitely many nonces (as a consequence of uniqueness), we typically have to consider infinitely many

[5]In our particular case, the environment could record the sender of each message that is received at any given round.

strands for a given protocol. Note that these strands do not necessarily represent computations of *different* agents. Indeed, it probably makes sense to consider them all as being performed by the same agent (but at most one of them being performed in a given execution of the protocol).

The bijection f in $\sqsubseteq_f$ is not needed if a different agent is associated with each strand. (That is, in this case it suffices to take f to be the identity.) Similarly, f is not needed if there is a bound k on the length of all strands in Σ. Indeed, it is needed only to take care of the possibility that there is an infinite sequence of strands, each intuitively a prefix of the next, and all associated with the same agent. For example, consider the strand space Σ where, again, $\Sigma = \{s_1, s_2, \dots\}$ but now $\mathrm{tr}(s_n) = \langle +u_1, \dots, +u_n \rangle$. Intuitively, in this strand space, s_n is a substrand of s_{n+1} (although, formally, there is no notion of substrand in strand spaces). Suppose that the mapping is such that $\mathcal{A}$ consists of one agent a_1 and A associates all the strands in Σ with a_1. If we did not allow such a map f (or, equivalently, required f to be the identity), then the only chains would be those of the form $B_0 \mapsto B_1 \mapsto \dots \mapsto B_k \mapsto B_k \mapsto B_k \mapsto \dots$ (for some finite k), where, for some strand s, each B_i is a prefix of s. If we apply our mapping to this collection of strands, in the resulting system, there is a single set of histories $V_{a_1} = \{\langle \mathsf{sent}(u_1) \rangle, \langle \mathsf{sent}(u_1), \mathsf{sent}(u_2) \rangle, \langle \mathsf{sent}(u_1), \mathsf{sent}(u_2), \mathsf{sent}(u_3) \rangle, \dots\}$, where each history in V_{a_1} is finite. However, the system generated by this set of histories contains an infinite run, which sends message u_i at time i. Unfortunately, there is no chain corresponding to this run. On the other hand, once we allow nontrivial bijections f, there is no problem. Abusing notation somewhat, there is a chain of the form $s_1 \mapsto s_2 \mapsto s_3 \mapsto \dots$ where a_1's history is unbounded, since $s_k \sqsubseteq_{f_k} s_{k+1}$, where f_k permutes s_k and s_{k+1} and is the identity on all other strands.

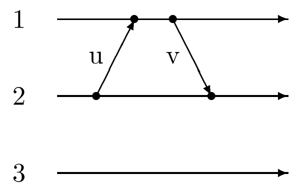
Intuitively, if f must be the identity, then every chain must "choose" the strand it is executing, which implicitly corresponds to choosing how many messages to send in that particular run. By providing a function f that permits us to "jump" to strands with the same prefix between any consecutive bundles of a chain, we are essentially modeling an agent that does not choose the length of the strand up front, but rather just performs the actions (and thus, if one strand is a prefix of another, it cannot tell which of the two strands it is performing).

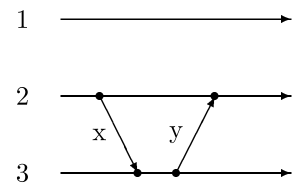
While it is important to recognize these subtleties, they do not arise in most protocols. For instance, strands for specific protocols will typically be of bounded length, and therefore the bijection f is not needed to define chains in the corresponding strand space.

4. TRANSLATING STRAND SYSTEMS TO STRAND SPACES

In this section, we consider the translation of strand systems into strand spaces. Specifically, given a strand system $\mathcal{R}$, is there a strand space which maps to $\mathcal{R}$ under a suitable agent assignment? In general, there is not. This result is not an artifact of our translation, but reflects a fundamental difference between strand spaces and strand systems. In particular, it does not depend on any of the subtleties that were pointed out at the end of last section.

To understand the difficulties, consider the following simple system $\mathcal{R}_1$. It essentially contains two runs r_1 and r_2, with distinct messages x, y, u, v:

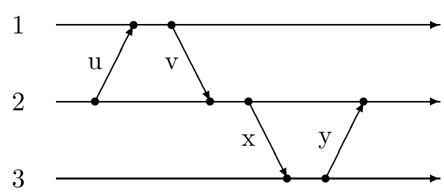


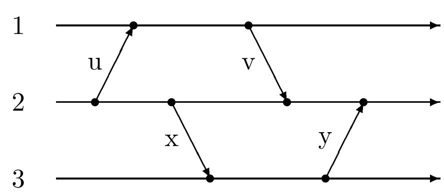


Because the MP1–3 assumptions on strand systems allow arbitrary delays between the events, there are more than two runs in the system; the essential fact is that, in any given run, agent 2 communicates only with agent 1 or only with agent 3. Formally, $\mathcal{R}_1$ is the strand system generated by taking:

$V_1 = \{\langle\rangle, \langle \mathsf{recv}(u)\rangle, \langle \mathsf{recv}(u), \mathsf{sent}(v)\rangle\}$,
$V_2 = \{\langle\rangle, \langle \mathsf{sent}(u)\rangle, \langle \mathsf{sent}(x)\rangle, \langle \mathsf{sent}(u), \mathsf{recv}(v)\rangle, \langle \mathsf{sent}(x), \mathsf{recv}(y)\rangle\}$, and
$V_3 = \{\langle\rangle, \langle \mathsf{recv}(x)\rangle, \langle \mathsf{recv}(x), \mathsf{sent}(y)\rangle\}$.

Under the mapping presented in the previous section, there does not exist a strand space that maps to this system, for any agent assignment. Intuitively, any strand space modeling the system $\mathcal{R}_1$ will need at least strands corresponding to runs r_1 and strands corresponding to runs r_2. Since these sets of strands do not interact (that is, they do not exchange any message), the translation of Section 3 will produce a system that contains runs that amount to all possible interleaving of the strands corresponding to r_1 and r_2. This results in a system that is strictly larger than $\mathcal{R}_1$. For example, it must contain runs with the following histories for agents 1, 2, and 3:





Roughly speaking, what is happening in the strand system is that agent 2 nondeterministically decides whether to send message u to agent 1 or message x to agent 3. In any run of the system, it sends one or the other, but not both. The problem here is that, in the strand-space framework, we cannot say "one or the other, but not both".

To make this precise, given an agent assignment A, define a translation T from strand spaces to strand systems to be *A-history preserving* if, given a strand space Σ,

—for each agent $a \in \mathcal{A}$, run $r \in T(\Sigma)$, and time m, there exists a bundle B in Σ such that the events in agent a's history $r_a(m)$ are exactly those that appear in nodes $\langle s, i\rangle$ in B such that $A(s) = a$;

—conversely, for each agent $a \in \mathcal{A}$ and bundle B of finite height in Σ, there exists a run $r \in T(\Sigma)$ and time m such that the events in agent a's history $r_a(m)$ are exactly those that appear in nodes $\langle s, i\rangle$ in B such that $A(s) = a$.

Notice that the translation T_A defined in the previous section is A-history preserving.

Theorem 4.1. *There is no agent assignment A and A-history preserving translation T from strand spaces to strand systems such that the strand system $\mathcal{R}_1$ is in the image of T.*

The example above suggests that in general, systems arising from an agent running a nondeterministic protocol may not be the image of a strand space under our translation. The problem in fact is more profound. Even if the agents are running deterministic protocols, the nondeterminism inherent in the delay of messages delivery may prevent a system from being the image of a strand space. Consider the following system with two agents. Agent 1 sends a message u to agent 2. If agent 2 hasn't received it yet, and hasn't sent a *nack* message yet, she sends a *nack*. When she gets message u, she sends an *ack*. Here, the strand space intuitively corresponding to this situation will include a strand for agent 1 where he sends u. For agent 2, we can consider at least the following two strands, $\langle -u, +ack\rangle$ and $\langle +nack, -u, +ack\rangle$. One can check that there exists a chain leading to the bundle made up of the following strand prefixes: $\langle +u\rangle$, $\langle -u, +ack\rangle$, and $\langle +nack\rangle$, leading, through our translation, to a possible history for agent agent of the form $\langle \mathsf{recv}(u), \mathsf{sent}(ack), \mathsf{sent}(nack)\rangle$, which does not arise in the original system. In this example, the problem does not occur because the agent makes a choice, but, intuitively, because the "environment" is making a choice when delivering messages. We will revisit these issues in Section 6, when we study the generation of systems from protocols.

5. EXTENDED STRAND SPACES

In the previous section, we showed that not all strand systems correspond to strand spaces. More precisely, we showed that some strand spaces could not be in the image of any history-preserving translation. How reasonable is the requirement that a translation be history preserving? Suppose that T is a translation from strand spaces to strand systems that is "acceptable" in some sense. It certainly seems reasonable to require that if $T(\Sigma) = \mathcal{R}$, then the events in every history $r_a(m)$ that arises in $\mathcal{R}$ correspond to events that agent a actually performed in some bundle. (Note that there is no need to consider infinite bundles here; if there is a bundle at all, it is finite.) Conversely, given a bundle B over Σ, it seems reasonable to require that there exists a history where a performs the same actions as it does in the bundle.

So exactly why is there no strand space corresponding to the system $\mathcal{R}_1$? Roughly speaking, given a strand space Σ, any set of strands that satisfies B1–4 is a bundle. Thus, once certain bundles exist, others are forced to exist too, including ones that do not correspond to any run in $\mathcal{R}_1$. For example, once there is a bundle corresponding to "2 sends u to 1 and gets a response v", and another bundle corresponding to "2 sends x to 3 and gets a response y", there has to be a bundle where 2 both sends u to 1 and sends x to 3. The strand-space framework cannot express "either this sequence of events happens or that one does, but not both". As we now show, this is essentially the only impediment standing in the way of a translation from strand spaces to strand systems. We extend the strand-space formalism with a notion of conflict that allows us to prohibit certain strands from appearing together in the same bundle, and then show that such extended strand spaces can model all strand systems.[6]

Define an *extended strand space* as a tuple $(\Sigma, \mathcal{A}, A, \mathit{Conf})$ consisting of a strand space Σ, a set $\mathcal{A}$ of agents, an agent assignment A from strands to agents, and a set $\mathit{Conf} = \{\mathit{Conf}_a : a \in \mathcal{A}\}$ of symmetric relations, indexed by agents, such that $\mathit{Conf}_a \subseteq A^{-1}(a) \times A^{-1}(a)$. The intuition is that if two strands s_1 and s_2 corresponding to the same agent a are such that $\mathit{Conf}_a(s_1, s_2)$, then s_1 and s_2 *conflict*; they cannot both appear in the same bundle of $(\Sigma, \mathcal{A}, A, \mathit{Conf})$. Formally, it is sufficient to refine the definition of a bundle. If $(\Sigma, \mathcal{A}, A, \mathit{Conf})$ is an extended strand space, a *bundle* B of $(\Sigma, \mathcal{A}, A, \mathit{Conf})$ is, as in the case of standard strand spaces, a subgraph $(\mathcal{N}_B, (\rightarrow_B \cup \Rightarrow_B))$ of the strand space Σ, satisfying B1–4 and, in addition:

B5. if $A(s_1) = A(s_2) = a$ and $\mathit{Conf}_a(s_1, s_2)$, then it is not the case that both B-height$(s_1) \geq 1$ and B-height$(s_2) \geq 1$ (intuitively, if $\mathit{Conf}_a(s_1, s_2)$, then s_1 and s_2 cannot both appear in B).

We can similarly define an infinite bundle as a subgraph satisfying B2–5; the notion of height remains unchanged.

Clearly, every bundle in an extended strand space $(\Sigma, \mathcal{A}, A, \mathit{Conf})$ is a bundle of Σ, since properties B1–4 still hold. Moreover, properties such as the penetrator bounds proved by THG carry over to extended strand spaces.

We now consider translations from extended strand spaces to strand systems and back. We first need to check that the construction of Section 3 that translates a strand space into a strand system applies to extended strand spaces. Since a bundle in an extended strand space is a bundle in the underlying strand space, we define the set $\mathit{Chains}(\Sigma, \mathcal{A}, A, \mathit{Conf})$ as the subset of $\mathit{Chains}(\Sigma, \mathcal{A}, A)$ where each chain is taken over bundles in the extended strand space. As in Section 3, we can associate a run r^C with every chain of $\mathit{Chains}(\Sigma, \mathcal{A}, A, \mathit{Conf})$, and we define $\mathcal{R}(\Sigma, \mathcal{A}, A, \mathit{Conf}) = \{r^C : C \in \mathit{Chains}(\Sigma, \mathcal{A}, A, \mathit{Conf})\}$. The analogue of Theorem 3.1 can be proved.

THEOREM 5.1. *$\mathcal{R}(\Sigma, \mathcal{A}, A, \mathit{Conf})$ is a strand system.*

Therefore, extended strand spaces can be translated into strand systems in such a way that chains correspond to the runs of the system. We abuse notation and call this family of translations T_A as well, where A is an agent assignment (although

[6]We do not want to imply that this is the only way to extend strand spaces to achieve this effect, nor do we claim that this approach is particularly original. Indeed, there is a vast literature in concurrency theory on the subject of implementing choice constructs in various formalisms; see, for instance, [Busi and Gorrieri 1994; Palamidessi 1997]. Independently, Crazzolara and Winskel [2001] have noticed the same deficiency in strand spaces, and have derived a similar notion of conflict between strands.

now the domain of T_A is extended strand spaces over the agent assignment A). However, the maps T_A are now onto, and the following theorem holds.

THEOREM 5.2. *Given a strand system $\mathcal{R}$ over $\mathcal{A}$, there exists an extended strand space $(\Sigma, \mathcal{A}, A, Conf)$ such that $T_A(\Sigma, \mathcal{A}, A, Conf) = \mathcal{R}$.*

Extending the strand space model with a notion of per-agent conflict relation is not the only way to extend the model to match the expressiveness of strand systems. For instance, it is possible to introduce a more general form of conflict specifying that an arbitrary pair of strands in a strand space cannot appear in any bundle. This notion of conflict does not require the introduction of agents in the strand-space framework. On the other hand, this extension is actually more expressive than strand systems as defined in this paper. For example, it is possible to say that a particular history of agent 1 and another history of agent 2 do not occur in the same run, something which cannot be done in a strand system. While it is straightforward to augment strand systems to capture this stronger notion of conflict, it is not clear that such a notion is of particular interest.

6. FROM PROTOCOLS TO SYSTEMS

Up until now, we have assumed that our strand spaces and systems were simply given. This is the assumption typically made in the strand spaces literature. In practice, however, strand spaces and systems arise out of the agents executing protocols. In this section, we review the basics of how to derive a strand system from an explicit protocol. This is a straightforward application of the techniques of [Fagin et al. 1995]. We then explore, using this approach, why the strand spaces approach is successful when dealing with typical protocols, despite the restrictions pointed out in Section 4. Roughly speaking, the strand systems that are generated from typical protocols are images of strand spaces via our translation. In other words, typical authentication protocols do not lead to systems that cannot be expressed as strand spaces; these protocols do not make choices. This result is not surprising given our previous discussion in Section 4, but it does formally ground our intuition. (On the other hand, we should point out that modern security protocols often involve choosing subprotocols.)

Intuitively, a protocol for agent $a \in \mathcal{A}$ is a description of what actions a may take as a function of her local state. What actions are we to allow in our protocols? This question ties in with the computational model implicitly assumed by strand spaces. Notice that in strand spaces, the receiver of a message cannot be specified. Indeed, there is an edge $n_1 \rightarrow n_2$ between all nodes of the form $+u$ and $-u$.[7] Therefore, we will consider a model where a send action sends a message nondeterministically to any agent. The only other action we allow beyond send actions is a "do nothing" no-op action. (We could also incorporate other actions, such as choosing keys, or tossing coins to randomize protocols.) For simplicity, we take actions to be deterministic, although protocols themselves can be nondeterministic. In other words, we will not consider an action such as "send some nondeterministically

[7] We could extend the notion of strand spaces to allow a specification of which $\rightarrow$ edges should be included; we could also add a "tagging" mechanism to messages. However, our interest here is not in extending the strand space formalism, but in modeling strand spaces as defined by THG.

chosen message u", but rather a protocol chooses nondeterministically among the actions "send u_1", "send u_2", etc.

We can formalize this intuition as follows. Fix a set L_a of local states for agent a (the local states that arise in some system) and a set ACT_a of possible actions that agent a can take. A *protocol* P_a for agent a is a function that associates with every local state in L_a a nonempty subset of actions in ACT_a. Intuitively, $P_a(\sigma)$ is the set of actions that agent a may perform in local state σ. Notice that a's actions can depend only on her local state. If P_a prescribes a unique action for a at each local state, then P_a is said to be *deterministic*. To consider the effect of all the agents' protocols on the global state of the system, define a *joint protocol* $\langle P_a : a \in \mathcal{A}\rangle$ to be a tuple consisting of a protocol for each of the agents. A joint protocol maps a global state to a set of *joint actions*, where a joint action is a tuple consisting of an action in ACT_a for each agent a. Define $\langle P_a : a \in \mathcal{A}\rangle(\langle \sigma_a : a \in \mathcal{A}\rangle) = \{\mathbf{a}_a : a \in \mathcal{A},\, \mathbf{a}_a \in P_a(\sigma_a)\}$.

Joint actions transform global states. Their effect is captured by a transition function τ mapping global states to sets of global states.[8] Given a joint protocol, a transition function, and a set of initial global states, we can generate a system in a straightforward way. Intuitively, the system consists of all the runs that are obtained by running the joint protocol from one of the initial global states. More formally, say that run r is *consistent with joint protocol P given transition function* τ if it could have been generated by P, that is, for all m, $r(m+1)$ is the result of applying a joint action $\mathbf{a}$ that could have been performed according to protocol P to $r(m)$. (More precisely, there exists a joint action $\mathbf{a} = \langle \mathbf{a}_a : a \in \mathcal{A}\rangle$ such that $\mathbf{a}_a \in P_a(r_a(m))$ and $r(m+1) \in \tau(\mathbf{a})(r(m))$.) For a joint protocol P, a transition function τ, and a set of initial state I, let $\mathcal{R}(P,\tau,I)$ be the set of all runs r consistent with P given τ such that $r(0) \in I$.

We saw in Section 2 that strand systems are asynchronous systems; they do not provide any guarantee either with respect to the time it takes to deliver a message, or with respect to the relative rates at which agents perform actions. We can capture this asynchrony by using an appropriate transition function. First, note that the only action in strand systems is that of sending a message. Also, the local states of an agent is its history. When agent a's component of a joint action $\mathbf{a}$ is a $\mathsf{send}(u)$ for some message u, $\tau(\mathbf{a})$ may or may not actually send the message. If the message is not sent, the agent's local state is unchanged, and hence the agent's protocol will allow the message to be re-sent. A message can be delivered in any round after it is sent (and may never be delivered at all). To capture this, we use the *strand transition function* τ_P of a joint protocol P, defined as follows: a global state $\langle \sigma'_a : a \in \mathcal{A}\rangle \in \tau_P(\langle \mathbf{a}_a : a \in \mathcal{A}\rangle)(\langle \sigma_a : a \in \mathcal{A}\rangle)$ iff for all $a \in \mathcal{A}$, σ'_a is either σ_a, $\sigma_a \cdot \mathsf{sent}(u)$ (only if $\mathbf{a}_a = \mathsf{send}(u)$), or $\sigma_a \cdot \mathsf{recv}(u)$ (only if $\mathsf{sent}(u) \in \sigma'_b$ for some b). We can check that using the strand transition function does indeed yield a strand system. Note that in a strand system, each agent is assumed to start with an empty initial local state. Hence, we always take the set of initial global state to be I_0, which contains only the empty global state.

[8]In [Fagin et al. 1995], the transition function is taken to be a function from global states to global states. The nondeterminism inherent in our definition is avoided by taking an *environment* as an extra agent in systems. For simplicity, we have not considered environments in this paper.

THEOREM 6.1. *$\mathcal{R}(P, \tau_P, I_0)$ is a strand system.*

We now have all the machinery to explain why strand spaces can be suitable for modeling typical protocols found in the literature. (We say *can* because part of the suitability issue depends on the actual properties we want to prove, as we will discuss in Section 7.) What do we mean by "suitable for modeling"? We have described above how protocols generate systems in a natural way. Furthermore, we saw in Section 4 that not every strand system arises as a translated strand space. In those cases, the most natural strand space allows bundles that do not correspond to states that actually occur in the system. A strand space is suitable as a model for a system if the translation of that strand space (under some agent assignment) yields the system. We interpreted our results of Section 4 as showing that strand spaces could not express choice (and other related forms of nondeterminism). Hence, intuition would indicate that if a protocol avoids such nondeterminism, strand spaces should be suitable for modeling the generated system.

It turns out that capturing this intuitive notion is not so easy. The restrictions that must be imposed to ensure that the generated strand system can be expressed as a strand space are nontrivial. To make them precise, we need a few definitions. For a class $\mathcal{P}$ of protocols, a joint protocol $\langle P_a : a \in \mathcal{A}\rangle$ is *decomposable* into protocols in $\mathcal{P}$ if for each agent a, we can find protocols $P_a^1, P_a^2, \ldots$ in $\mathcal{P}$ such that for all global states $\langle \sigma_a : a \in \mathcal{A}\rangle$ we have $P_a(\sigma_a) = \cup_i P_a^i(\sigma_a)$. In other words, a decomposable joint protocol can be understood as each agent running a set of protocols in a given class $\mathcal{P}$. A deterministic protocol P is *monotone* if there exists events $e_1, e_2, \ldots$ (the sequence may be finite or infinite), such that for any local state σ, we have $P(\sigma) = \{\mathsf{send}(u)\}$, if $e_{i+1} = \mathsf{sent}(u)$ and i is the largest index such that $e_1, \ldots, e_i \in \sigma$; and $P(\sigma) = \{\mathsf{no\text{-}op}\}$ otherwise. Informally, a deterministic protocol is monotone if the possible action at a state depends only on whether or not a given set of events has occurred. (Other events in the state does not affect the possible action.) For example, a monotone protocol may be of the form: send message u_1, wait for message u_2, send message u_3. This is monotone in our sense, since u_1 is sent not matter what, and u_3 is sent if and only if u_2 is received. This means that if certain messages are sent by a in one run of the protocol, they must be sent by a in all runs of the protocol. This must be true even if the protocol is run in parallel with other protocols (or other instantiations of the same protocol). In a sense, monotone protocols don't "interact"; if an agent is running multiple monotone protocols at the same time, they cannot keep each other from proceeding.

It is not hard to show that neither of the two systems that we showed in Section 4 were not representable as strand spaces are generated by monotone protocols. The first protocol (where there was a nondeterministic choice) is not itself monotone, since agent 2 nondeterministically chooses to send a message to agent 1 or agent 3. Nor can it be split into two deterministic monotone protocols, one to communicate with agent 1 and one to communicate with agent 3. Sending a message in one protocol would prevent a message being sent in another, and thus neither of the two deterministic protocols is monotone. Intuitively, the two protocols interact, something disallowed by monotonicity. Agent 2's protocol in the *nack*/*ack* example is not monotone either. The obvious sequence of events for agent 2's protocol is $\mathsf{sent}(nack), \mathsf{recv}(u), \mathsf{sent}(ack)$, but this does capture the protocol, since in runs where 2 receives u before sending the *nack* message will not arise in the protocol corresponding to this sequence.

The following theorem shows that joint protocols decomposable into monotone protocols can indeed be modeled by strand spaces:

THEOREM 6.2. *If P is a joint protocol decomposable into monotone protocols, then there exists a strand space Σ and an agent assignment A such that $T_A(\Sigma) = \mathcal{R}(P, \tau_P, I_0)$.*

Theorem 6.2 somewhat explains why the restrictions on the modeling power of strand spaces we pointed out in Section 4 are not an issue when analyzing security protocols of the kind typically found in the literature. These protocols are monotone. Note that the penetrator in strand spaces analyses also runs a protocol that is a union of monotone protocols: send a new message, send a concatenation of two received messages, send part of a received message. These monotone protocols correspond exactly to the so-called penetrator strands.

7. DISCUSSION

In this paper, we have investigated the relationship between strand spaces and multi-agent systems. Our results show that strand spaces are strictly less expressive than strand systems, a subclass of multi-agent systems that seems to capture the assumptions underlying strand spaces, in two quite distinct respects. The first is that strand spaces cannot express choice, the fact that exactly one of two possible behaviors is chosen. The second is that strand spaces have no notion of agents.

How serious are these two issues? That depends, of course, on what we are trying to prove. Consider first the inability of strand spaces to express choice. In [Thayer et al. 1999b], the types of properties proved typically have the form "for all bundles in the strand space, X happens". One way to interpret our result of Section 4 is that when a strand space is used to model a system, some of the bundles may not correspond to situations that actually arise in the system—those bundles can be seen as "impossible" bundles. This is not a problem, of course, if the property of interest in fact holds in the larger system. However, this may not always be the case. For example, we may well want to prove that a property like "agent 2 sends at most one message" holds in all executions of a protocol. If the protocol also has the property that agent 2 can send messages to either 1 or 3 (as is the case in the protocol described by the system $\mathcal{R}_1$ in Section 4), then the fact that agent 2 sends at most one message in every execution of the protocol will simply not be provable in the strand-space framework. On the other hand, as we saw in Section 6, if we consider only strand systems generated from protocols decomposable into monotone protocols, a fairly restrictive class of protocols, then we know that there is a strand space modeling the system that does not contain any such "impossible" bundles.

The runs of a strand system can be viewed as a linearization of bundles, that is, an explicit ordering of the actions performed by agents in different bundles. THG suggest that results about strands can be imported to runs. For example, on page 226, they say "[Alternatively,] results about authentication protocols proved in a strand space context can be imported into the more usual linear models by linearizing the bundles." Our results point to subtleties in doing this. More precisely, while results about strands can be imported to results about runs (the runs that arise from translating the strand space to a system), the converse may not be true, depending on the expressiveness of the language.

Turning to the issue of agents, the strand-space framework assumes that messages relayed between strands form the only means of exchanging information between strands. In other words, there is no shared state between strands. Therefore, for all intents and purposes, we can imagine that every strand is executed by a different agent. On the other hand, if the same agent is executing two strands then, intuitively, it should know whatever is happening on both strands, without requiring communication between them. Furthermore, as soon as one wants to analyze the properties of strand spaces using belief and knowledge, agents to which the knowledge can be ascribed are needed. But even without bringing in knowledge, we need to be careful in interpreting security results proved under the assumption that different agents perform different strands. Clearly this assumption is not, in general, true. Ideally, security protocols should be proved correct under any "reasonable" assignment of agents to roles in the security protocol. At the very least it should be clear under which assignments the result holds. For instance, it is known that methods for the analysis of cryptographic protocols that fail to handle multiple roles for the same agent do not yield dependable results, as they may not reveal *multi-role flaws*. Snekkenes [1992] studies such flaws in the context of various cryptographic protocol logics. Multi-role flaws commonly arise when a cryptographic protocol logic implicitly assumes that if an agent a takes on a role A in some session, then he will not also take on another role B in some different session. This assumption is often a consequence of the identification of the notions of role and agent. Snekkenes shows that reasonable protocols that can be proved correct under the assumption that an agent takes on the same role in all sessions are flawed if this assumption is dropped. Recent work on analyzing mixed protocols using strand spaces [Thayer et al. 1999a] shows that strand spaces can be extended to deal with what essentially amount to multi-role flaws. However, the approach often requires *phantom messages* (messages that are not actually exchanged during runs of the protocols) to carry state information between the different protocol strands corresponding to the same agent.

Some of the topics we have explored in this paper appear in various forms in other work. For example, Cervesato *et al.* [2000] define a notion of *parametric strand*, essentially a strand where messages may contain variables. Parameterized strands correspond to roles, which are implicit in the original work on strand spaces. The work of Cervesato *et al.* also deals with the evolution of the system described by a strand space; they define a one-step transition between bundles. The transition is reminiscent of the one we describe in Section 3, but is restricted to extending a single strand at a time. (They also allow actions specific to their formalization, such as the instantiation of a strand from a parametric strand.)

The set of runs in the system and the agent assignment are particularly significant when we consider specifications that are not *run-based* [Fagin et al. 1995; Halpern 2000]. A run-based specification is checked on a per-run basis. For example, "agent 2 sends at most 1 message" is a run-based specification: given a run, one can check whether the property holds for that run. A run-based specification holds for a set of runs if it holds for all runs in the set. In contrast, a *knowledge-based specification*

[Fagin et al. 1995; Halpern 2000] such as "after running the protocol, agent 2 knows X" cannot be checked on a per-run basis, as it relies on the set of runs *as a whole* to verify the property. It holds if, in all runs in the system that agent 2 considers possible after running the protocol, X holds. Clearly it does not suffice to look at an individual run to determine whether such a property holds. Similarly, probabilistic specifications like "X holds in at most 3% of the runs" also depend on the whole system and cannot be checked simply by examining individual runs.

Typical specifications in the security literature are safety properties (in the sense of Alpern and Schneider [1985], "bad things don't happen"), and hence are run-based. Run-based specifications have the property that if they hold in a system, they hold in any subset of the runs of the system. It is "safe" to prove that a run-based specification holds of a strand space which translates to a superset of the intended system. Proving that the property holds for "impossible" runs does not hurt. This is not the case for properties that are not run-based. We believe that knowledge-based specifications, as well as probabilistic ones, will play a significant role in the design and analysis of security protocols. Fairness is a good example. A protocol is *fair* if intuitively no protocol participant can gain an advantage over other participants by misbehaving. In the context of fair exchange protocols [Asokan et al. 1998; Ben-Or et al. 1990; Shmatikov and Mitchell 2000], where two agents exchange one item for another, fairness ensures that either each agent receives the item it expects, or neither receives any information about the other's item. This notion of "not receiving any information" can be interpreted as meaning that no knowledge is gained. Our results suggest that strand spaces, as currently defined, will have difficulty handling such specifications.

We should point out that it is straightforward to reason about knowledge in the context of strand spaces. For instance, Syverson [1999] describes a framework where the set of bundles in a strand space is viewed as a set of possible worlds. He associates with every strand in the strand space a principal, as we do, and uses this setting to provide a model for the knowledge of principals. As his framework is directly based on strand spaces, it suffers from the same expressiveness problems we pointed out in Section 4. This emphasizes that the problem we point out is not a problem of how to express knowledge in strand spaces. Rather, it is purely a problem with expressiveness of the *models* allowed in the strand-space framework.

Despite these criticisms, we feel strand spaces are an important and useful formalism. They can be used to provide simple, transparent proofs of run-based properties. Our results suggest it is worth exploring their limitations and the extent to which extensions of strand spaces (such as the extended strand spaces introduced here) retain these properties.

A. PROOFS

THEOREM 3.1. *$\mathcal{R}(\Sigma, \mathcal{A}, A)$ is a strand system.*

PROOF. Let V_a consist of all the histories $r_a(m)$ for $r \in \mathcal{R}(\Sigma, \mathcal{A}, A)$. Let $\mathcal{R}'$ be the strand system generated by the sequence $\langle V_a : a \in \mathcal{A} \rangle$. To show that $\mathcal{R}(\Sigma, \mathcal{A}, A)$ is a strand system, it clearly suffices to show that $\mathcal{R}(\Sigma, \mathcal{A}, A) = \mathcal{R}'$. It is easy to check from the construction that every run in $\mathcal{R}(\Sigma, \mathcal{A}, A)$ satisfies MP1–3, and thus is in $\mathcal{R}'$. This shows that $\mathcal{R}(\Sigma, \mathcal{A}, A) \subseteq \mathcal{R}'$.

To show that $\mathcal{R}' \subseteq \mathcal{R}(\Sigma, \mathcal{A}, A)$, let r be a run in $\mathcal{R}'$. We know that r satisfies MP1–3, and that $r_a(m) \in V_a$ for all $m \geq 0$. We need to construct a chain C such that $r_a(m) = \mathit{hist}_a^m(C)$ for all $a \in \mathcal{A}$. Unfortunately, we cannot simply construct the chain inductively, bundle by bundle. While this would work if different strands were associated with different agents, in general, making the correct choice of strands at each step (correct in the sense that the construction will not get stuck at a later point) turns out to require arbitrary lookahead into the run. Roughly speaking, this is because it is not clear which combination of strands for agent a to choose to make up a's local state in a particular bundle.

Instead, we proceed as follows. Intuitively, we want to determine for each agent which strand prefix to extend at every step of the chain. Once we have found for each agent an appropriate way of extending strand prefixes at every step, it is not hard to construct the bundles in the chain.

We start with some definitions. Given a node $\langle s, k\rangle$ in Σ, let $\mathrm{tr}(s, k)$ be the prefix of $\mathrm{tr}(s)$ of length k. Given a bundle B and an agent a, let

$$\mathrm{Tr}_a(B) = \{\{\mathrm{tr}(s, k) : \langle s, k\rangle \in \mathcal{N}_B,\ \langle s, k+1\rangle \notin \mathcal{N}_B,\ k \geq 1,\ A(s) = a\}\},$$

where we use the $\{\{\}\}$ notation to denote multisets. Thus, $\mathrm{Tr}_a(B)$ is the multiset consisting of all the maximal prefixes of strands associated with a having at least one node in B. Note that $\mathrm{Tr}_a(B)$ is a multiset, not a set. It is quite possible that there are distinct nodes $\langle s, k\rangle$ and $\langle s', k\rangle$ in $\mathcal{N}_B$ such that $\mathrm{tr}(s, k) = \mathrm{tr}(s', k)$ and $\langle s, k+1\rangle, \langle s', k+1\rangle \notin B$. In this case, $\mathrm{tr}(s, k)$ is listed at least twice in the multiset. Given a multiset M of sequences, let $\mathcal{B}_a(M) = \{B : \mathrm{Tr}_a(B) = M\}$. That is, $\mathcal{B}_a(M)$ consists of all bundles where the actions performed are precisely those specified by the sequences in M.

For each agent a, we inductively construct the following tree, whose vertices are labeled by multisets of sequences. The root is labeled by the empty multiset. Suppose a vertex u at level m (that is, at distance m from the root) is labeled with the multiset M. If $r_a(m + 1) = r_a(m)$, then u has a unique successor, also labeled with M. If, on the other hand, $r_a(m + 1) = r_a(m) \cdot e$ for some event e, then let t be the term corresponding to e (i.e., if e is $\mathsf{sent}(u)$ then t is $+u$, and if e is $\mathsf{recv}(u)$ then t is $-u$). For each sequence S in M, let M_S be the multiset that results from replacing S in M by $S \cdot t$. We construct a successor of M labeled M_S if $\mathcal{B}_a(M_S) \neq \emptyset$. (If $\mathcal{B}_a(M_S) \neq \emptyset$ and there are several occurrences of S in M, then we construct one successor for each occurrence.) In addition, if $\mathcal{B}_a(M \cup \{\{\langle t\rangle\}\}) \neq \emptyset$, we construct a successor of u labeled $M \cup \{\{\langle t\rangle\}\}$. Note that, for all multisets labeling a level-m vertex, the set of events specified by the sequences in M are precisely those performed in $r_a(m)$.

Our goal is to find an infinite path in this tree. That such a path exists is immediate from König's Lemma, once we show that the tree has an infinitely many vertices, each with finite outdegree.

An easy induction shows that a multiset at level m has at most m elements (counted with multiplicity). Moreover, it is immediate from the construction that the outdegree of a vertex on the tree is at most one more than the cardinality of the multiset labeling it. Thus, it follows that the outdegree of each vertex is finite.

Showing that the tree has an infinite number of vertices is also relatively straightforward. We show by induction on m that for all times m, if $r_a(m) = \mathit{hist}_a^k(C)$

and $C = B_0 \mapsto B_1 \mapsto \dots$, then there is a vertex at level m in the tree labeled by the multiset $\mathrm{Tr}_a(B_k)$. The base case is immediate, since $\mathrm{Tr}_a(\emptyset) = \{\{\}\}$ is the label of the root of the tree. Now suppose that the result holds for m; we prove it for $m+1$. Suppose that $r_a(m+1) = \mathit{hist}_a^k(C)$. Then there must be some $k' \leq k$ such that $r_a(m) = \mathit{hist}_a^{k'}(C)$. Moreover, either $\mathit{hist}_a^{k'}(C) = \mathit{hist}_a^k(C)$, in which case $r_a(m) = r_a(m+1)$, or $\mathit{hist}_a^k(C)$ is the result of appending one event, say e, to $\mathit{hist}_a^{k'}(C)$ and $r_a(m+1)$ is the result of appending e to $r_a(m)$. If $C = B_0 \mapsto B_1 \mapsto \dots$ then, by the induction hypothesis, there is a vertex u of the tree at level m labeled by $M = \mathrm{Tr}_a(B_{k'})$. If $r_a(m) = r_a(m+1)$, then $M = \mathrm{Tr}_a(B_k)$ is also the label of a successor of u. Otherwise, if $M' = \mathrm{Tr}_a(B_k)$, it is clear that M' is the result of extending one of the strands in M by one node (corresponding to event e). Thus, M' is the label of some successor of u. This completes the inductive step. Since r is in $\mathcal{R}'$, it follows that, for all m, there exists some chain C and k such that $r_a(m) = \mathit{hist}_a^k(C)$. Thus, there are infinitely many vertices in the tree.

It now follows from König's Lemma that there is an infinite path in the tree. Thus, it follows that, for every agent a, there exists an infinite sequence $M_0^a, M_1^a, \dots$ of multisets, such that $\mathcal{B}_a(M_k^a) \neq \emptyset$ for all k. We now construct a chain $C = B_0 \mapsto B_1 \mapsto \dots$, by building the bundle B_k from the traces in $\{M_k^a : a \in \mathcal{A}\}$. For each a and k, there is a bundle B_k^a such that $\mathrm{Tr}_a(B_k^a) = M_k^a$. Let B_k consists of the nodes in $\cup_{a \in \mathcal{A}} B_k^a$ (so that the strands associated with a in B_k are precisely those associated with a in B_k^a), adding $\rightarrow$ edges between corresponding nodes according to MP2 in the run r. That B_k is a bundle follows from the fact that every node appearing in a multiset M_k^a corresponds to an event in $r_a(k)$, by construction. It should be clear that for all k, $B_k \mapsto B_{k+1}$, since for each agent, the traces are extended by a single node, and we can pick the bijection f to map strands from B_k to B_{k+1} so that the corresponding sequences in M_k^a and M_{k+1}^a match.

A straightforward induction argument shows that the chain $C = B_0 \mapsto B_1 \mapsto \dots$ is such that $r_a(m) = \mathit{hist}_a^m(C)$ for all $m \geq 0$. $\square$

THEOREM 3.2. *Every global state of $\mathcal{R}(\Sigma, \Sigma, id)$ is message-equivalent to a bundle of Σ of finite height, and every bundle of Σ of finite height is message-equivalent to a global state of $\mathcal{R}(\Sigma, \Sigma, id)$.*

We first prove two lemmas about chains.

LEMMA A.1. *In a chain $C = B_0 \mapsto B_1 \mapsto B_2 \mapsto \dots$, the height of B_n is at most $2n$.*

PROOF. We show this by induction on n. Clearly, the height of B_0 is 0. Assume the result holds for the bundle B_m. Consider the bundle B_{m+1}. Since $B_m \mapsto B_{m+1}$, there is a bijection f such that $B_m \sqsubseteq_f B_{m+1}$. Consider a causal path $n_1 \rightsquigarrow n_2 \rightsquigarrow \dots$ in B_{m+1}, where $\rightsquigarrow$ is either $\rightarrow$ or $\Rightarrow$. We claim that it contains at most two "new nodes", that is, it contains at most two nodes in B_{m+1} not of the form $\langle f(s), i\rangle$ for some node $\langle s, i\rangle$ in B_m; moreover, the "new" nodes must come at the end of the causal path. To see this, suppose that n is a new node on the path and $n \rightsquigarrow n'$ for some n' on the path. If n' is not a new node, it cannot be the case that $n \rightarrow n'$ (for otherwise, by B2, n would not be a new node), and it cannot be the case that $n \Rightarrow n'$ (for otherwise, by B3, n would not be a new node). Thus, n' must be a new node. It follows that all the new nodes on the causal path must follow the old nodes on the path. Now suppose that there are three new nodes on the path; then it must be the case that there are three new nodes n, n', n'' such that $n \leadsto n' \leadsto n''$. It cannot be the case that $n \Rightarrow n'$, for then n and n' are both on the same strand, contradicting the assumption in the construction that at most one new event is added per agent. Similarly, it cannot be the case that $n' \Rightarrow n''$. Thus, we must have $n \rightarrow n' \rightarrow n''$. But then $\mathrm{term}(n') = -u$ for some message u, and it cannot be the case that $n' \rightarrow n''$. Thus, it follows that the causal path has at most two new nodes. Since, by the induction hypothesis, there are at most $2m+1$ "old" nodes on the path, the path has at most $2m+3$ nodes and hence length at most $2m+2$, as desired. $\square$

Note that Lemma A.1 does not depend on the assumption that each strand is associated with a distinct agent; the following lemma does.

LEMMA A.2. *If B is bundle of finite height, then there exists bundles $B_1, \ldots, B_k$ for some k such that $B_0 \mapsto B_1 \mapsto \ldots \mapsto B_k \mapsto B$.*

PROOF. First note that if n is the last node on a causal path in a bundle B of maximum length, then either $\mathrm{term}(n) = -u$ for some u or $\mathrm{term}(n) = +u$ for some u and there is no corresponding receive node in B.

We now prove the result by induction on the height of B, that is the length of the longest causal path. Clearly, if the height of B is 0, then $B = B_0$. Otherwise, let B' be the bundle derived from B in the following way: for every strand $s \in \Sigma$, if the last term of the prefix of s in B is $-u$ for some u or if the last term is $+u$ and there is no corresponding $-u$ in B, then let B' contain the prefix of s that consists of every node in s that is in B but the last one; otherwise, let B' contain the same prefix of s as B. Clearly, $B' \mapsto B$. (Here we need the assumption that each strand is associated with a different agent to ensure that in going from B' to B, each agent performs at most one action.) Moreover, by the initial observation, B' does not include the last node of any causal path of maximum length in B. Therefore, the height of B' is one less than the height of B. Applying the induction hypothesis, we get bundles $B_0 \mapsto B_1 \mapsto \ldots \mapsto B_k \mapsto B' \mapsto B$, proving the result. $\square$

PROOF. (Theorem 3.2) If $\langle \sigma_s : s \in \Sigma \rangle$ is a global state in $\mathcal{R}(\Sigma, \Sigma, id)$, then there must be some chain $C = B_0 \mapsto B_1 \mapsto \ldots$ and time m such that $r^C(m) = \langle \sigma_s : s \in \Sigma \rangle$. By construction, $r^C_s(m) = hist^m_s(C)$, for each strand $s \in \Sigma$. (Recall that $\mathcal{A} = \Sigma$; we are associating each strand with a different agent.) Moreover, $hist^m_s(C)$ is just the sequence of events performed in strand s in B_m (that is, the prefix of $\mathrm{tr}(s)$ in B_m, under the standard correspondence between terms and events). Therefore, $\langle \sigma_s : s \in \Sigma \rangle$ is message-equivalent to B_m. Moreover, by Lemma A.1, B_m has finite height.

Conversely, given a bundle B of finite height, by Lemma A.2, there must exist m and bundles $B_0, \ldots, B_m$ such that $B_0 \mapsto \ldots \mapsto B_m \mapsto B$. Thus, $C = B_0 \mapsto \ldots \mapsto B_m \mapsto B \mapsto B \mapsto B \mapsto \ldots$ is a chain. Let r^C be the run in $\mathcal{R}(\Sigma, \Sigma, id)$ corresponding to C. By the same argument as above, $r^C(m+1)$ is message-equivalent to B. $\square$

THEOREM 4.1. *There is no agent assignment A and A-history preserving translation T from strand spaces to strand systems such that the strand system $\mathcal{R}_1$ is in the image of T.*

Proof. By way of contradiction, suppose that Σ is a strand space, A is an agent assignment, T is a translation which is A-history preserving, and $T(\Sigma) = \mathcal{R}_1$. Since T is A-history preserving, the presence of r_1 ensures that there is a bundle B_1 in Σ such that associated with agent 2 in B_1 is either a strand with prefix $\langle +u, -v\rangle$ or strands with prefix $\langle +u\rangle$ and $\langle -v\rangle$, and associated with agent 1 in B_1 there is either a strand with prefix $\langle -u, +v\rangle$ or strands with prefix $\langle -u\rangle$ and $\langle +v\rangle$. Similarly, the presence of r_2 in $\mathcal{R}_1$ guarantees that there is a bundle B_2 in Σ such that associated with agent 2 in B_2 is either a strand with prefix $\langle +x, -y\rangle$ or strands with prefix $\langle +x\rangle$ and $\langle -y\rangle$, and associated with agent 3 is either a strand with prefix $\langle -x, +y\rangle$ or strands with prefix $\langle -x\rangle$ and $\langle +y\rangle$. In all those cases, there must be a bundle containing nodes with the terms $+u$, $-u$, $+v$, $-v$, $+x$, $-x$, $+y$, and $-y$. The nodes $+u$, $-v$, $+x$, and $-y$ are all on strands associated with agent 2. Since T is A-history preserving, there must be a run in $\mathcal{R}_1$ that contains four events for agent 2. This is a contradiction. $\square$

Theorem 5.1. *$\mathcal{R}(\Sigma, \mathcal{A}, A, \mathit{Conf})$ is a strand system.*

Proof. The proof is similar to that of Theorem 3.1. We simply need to check that when we are proving the $\mathcal{R}' \subseteq \mathcal{R}(\Sigma, \mathcal{A}, A, \mathit{Conf})$ inclusion and constructing each bundle B_k in the chain C from the collection of traces $\{M^a_k : a \in \mathcal{A}\}$, each bundle is in fact a bundle in the extended strand space sense. This follows from the fact that we can choose for each agent a the strands making up the bundle in such a way that none of the strands conflict, since we assumed that $\mathcal{B}_a(M^a_k) \neq \emptyset$ for M^a_k, and therefore there must exist strands with the appropriate prefixes that do not conflict. $\square$

Theorem 5.2. *Given a strand system $\mathcal{R}$ over $\mathcal{A}$, there exists an extended strand space $(\Sigma, \mathcal{A}, A, \mathit{Conf})$ such that $T_A(\Sigma, \mathcal{A}, A, \mathit{Conf}) = \mathcal{R}$.*

Proof. Let V_a be a set of histories for each agent a, such that $\mathcal{R}$ is generated by the sequence $\langle V_a : a \in \mathcal{A}\rangle$. Without loss of generality, assume that each V_a is minimal, in the sense that every history in V_a actually appears in some run of $\mathcal{R}$. Define the strand space $\Sigma = \{s^h_a : a \in \mathcal{A}, h \in V_a\}$ with a trace mapping $\mathrm{tr}(s^{\langle e_1,\dots,e_k\rangle}_a) = \langle t_1, \dots, t_k\rangle$, where if e_i is $\mathsf{sent}(u)$, then t_i is $+u$, and if e_i is $\mathsf{recv}(u)$, then t_i is $-u$.

We define the conflict relation $\mathit{Conf} \subseteq \Sigma \times \Sigma$ to ensure that bundles include only one strand per agent. We set $\mathit{Conf}(s^h_a, s^{h'}_a)$ if and only if $h \neq h'$. Intuitively, since a bundle in $(\Sigma, \mathcal{A}, A, \mathit{Conf})$ can include only one strand per agent, and since strands correspond to possible local states, bundles correspond to global states of the system $\mathcal{R}$.

We show that T_A maps $(\Sigma, \mathcal{A}, A, \mathit{Conf})$ to $\mathcal{R}$, via the agent assignment $A(s^h_a) = a$. This is a direct consequence of the proof of Theorem 5.1. We know that $\mathcal{R}$ is generated by $\langle V_a : a \in \mathcal{A}\rangle$. We also know that $T_A(\Sigma, \mathcal{A}, A, \mathit{Conf})$ is generated by $\langle V'_a : a \in \mathcal{A}\rangle$, where $V'_a = \{\mathit{hist}^m_a(C) : C \in \mathit{Chains}(\Sigma, \mathcal{A}, A, \mathit{Conf}), m \geq 0\}$. Therefore, to show that $T_A(\Sigma, \mathcal{A}, A, \mathit{Conf}) = \mathcal{R}$, it is sufficient to show that $V_a = V'_a$ for all $a \in \mathcal{A}$.

Fix an agent $a \in \mathcal{A}$. We first show that $V_a \subseteq V'_a$. Let h be a history in V_a, and let $r \in \mathcal{R}$ and $m \geq 0$ be such that $r_a(m) = h$. For each $k \leq m$, define B_k to be the bundle formed by the strands $\{s_a^{r_a(k)} : a \in \mathcal{A}\}$, with edges between nodes on different strands given by MP2. (That B_k is a bundle follows from the properties MP1–3 on r.) It is easy to see that $B_k \mapsto B_{k+1}$ (for $k = 0, \ldots, m-1$). Let C be the chain $B_0 \mapsto \ldots \mapsto B_m \mapsto B_m \mapsto B_m \mapsto \ldots$. Then $hist_a^m(C)$ is just the set of events corresponding to strand $s_a^{r_a(m)} = s_a^h$, which is just h. Therefore, $h \in V'_a$. Showing that $V'_a \subseteq V_a$, is similar. Let h be a history in V'_a, so that there exists a chain C with $h = hist_a^m(C)$ for some $m \geq 0$. By construction, there exists a run $r^C \in \mathcal{R}$ such that $r_a^C(m) = hist_a^m(C) = h$. Thus, h is a local state of some run in $\mathcal{R}$, and $h \in V_a$. $\square$

THEOREM 6.1. *$\mathcal{R}(P, \tau_P, I_0)$ is a strand system.*

PROOF. Let $\mathcal{R}$ be $\mathcal{R}(P, \tau_P, I_0)$, and let V_a consists of all the histories $r_a(m)$ for $r \in \mathcal{R}$. Let $\mathcal{R}'$ be the strand system generated by $\langle V_a : a \in \mathcal{A}\rangle$. It is sufficient to show that for all runs r, $r \in \mathcal{R}$ iff $r \in \mathcal{R}'$.

First, assume that $r \in \mathcal{R}$, that is, that r is consistent with P given τ_P, and that $r(0) \in I_0$. By construction, $r_a(m) \in V_a$ for all a and all m, and hence r satisfies MP1. By definition of τ_P, if $\mathsf{recv}(u) \in r_a(m)$, then $\mathsf{sent}(u) \in r_b(m)$ for some b, and hence r satisfies MP2. Finally, $r_a(0)$ is the empty sequence because $r(0) \in I_0$, and by definition of τ_P, $r_a(m+1)$ is either $r_a(m)$ or the result of appending one event to $r_a(m)$, and hence r satisfies MP3. Therefore, $r \in \mathcal{R}'$.

Second, assume that $r \in \mathcal{R}'$, that is, r satisfies MP1–3. The fact that $r(0) \in I_0$ is a consequence of MP3: $r_a(0)$ is the empty sequence for all a. To show consistency with P given τ_P, we exhibit, for any m, a joint action $\mathbf{a}$ such that $r(m+1) \in \tau_P(\mathbf{a})(r(m))$. Let $r(m) = \langle \sigma_a : a \in \mathcal{A}\rangle$, and $r(m+1) = \langle \sigma'_a : a \in \mathcal{A}\rangle$. For any $a \in \mathcal{A}$, if $\sigma'_a = \sigma_a$, let $\mathbf{a}_a = \mathsf{no\text{-}op}$; if $\sigma'_a = \sigma_a \cdot \mathsf{sent}(u)$, let $\mathbf{a}_a = \mathsf{send}(u)$; if $\sigma'_a = \sigma_a \cdot \mathsf{recv}(u)$, let $\mathbf{a}_a = \mathsf{no\text{-}op}$ (by MP1 and MP2, we know that there must exist a $\mathsf{sent}(u)$ in σ'_b for some b). We can check that $\mathbf{a} = \langle \mathbf{a}_a : a \in \mathcal{A}\rangle$ has the required property. Hence, r is consistent with P given τ_P, and $r(0) \in I_0$, and therefore $r \in \mathcal{R}$. $\square$

THEOREM 6.2. *If P is a joint protocol decomposable into monotone protocols, then there exists a strand space Σ and an agent assignment A such that $T_A(\Sigma) = \mathcal{R}(P, \tau_P, I_0)$.*

PROOF. By definition, for all agents $a \in \mathcal{A}$, there exist monotone protocols $P_a^1, P_a^2, \ldots$. For each such protocol P_a^i, we can find events $e_{a,1}^i, e_{a,2}^i, \ldots$ as in the definition. Let $|P_a^i|$ denote the length of this sequence (which could be ∞).

Construct the strand space $\Sigma = \{s_a^{i,n} : a \in \mathcal{A}, i \geq 1, 1 \leq n \leq |P_a^i|\} \cup \{s_a^u : a \in \mathcal{A}, u \in M\}$. The strand $s_a^{i,n}$ corresponds to a prefix of length n of the events in the sequence for P_a^i. More precisely, its trace mapping is a trace mapping $\mathrm{tr}(s_a^{i,n}) = \langle t_1, \ldots, t_n\rangle$, where for all $1 \leq j \leq n$, if $e_{a,j}^i = \mathsf{sent}(u)$, then t_j is $+u$, and if $e_{a,j}^i = \mathsf{recv}(u)$, then t_j is $-u$. The strands of the form s_a^u are simple strands corresponding to receiving message u; there is one such strand for each message $u \in M$, and for each agent $a \in \mathcal{A}$. The trace mapping is simply $\mathrm{tr}(s_a^u) = \langle -u\rangle$. (These strands will be used to account for unsolicited messages delivered to agent a.) The agent assignment A is simply defined by taking $A(s_a^{i,n}) = a$ and $A(s_a^u) = a$, as expected.

Recall from Section 3 that $T_A(\Sigma, \mathcal{A}, A)$ maps to the set of runs $\{r^C : C \in \mathit{Chains}(\Sigma, \mathcal{A}, A)\}$. We show that this set of runs is just $\mathcal{R}(P, \tau_P, I_0)$.

First, let C be a chain in $\mathit{Chains}(\Sigma, \mathcal{A}, A)$. Recall that r^C is the run with $r^C(m) = \langle \mathit{hist}_a^m(C) : a \in \mathcal{A}\rangle$. To show that r^C is in $\mathcal{R}(P, \tau_P, I_0)$, it suffices to show that r^C is consistent with P given τ_P, and that $r^C(0) \in I_0$. The latter is an immediate consequence of the fact that $r_a^C(0) = \mathit{hist}_a^0(C) = \langle\rangle$. The former requires showing that for all m, we can find a joint action $\mathbf{a} = \langle \mathbf{a}_a : a \in \mathcal{A}\rangle$ such that $\mathbf{a}_a \in P_a(r_a^C(m))$ and $r^C(m+1) \in \tau_P(\mathbf{a})(r^C(m))$. Fix m. For an agent $a \in \mathcal{A}$, if $\mathit{hist}_a^m(C) = \mathit{hist}_a^{m+1}(C)$, take $\mathbf{a}_a = \mathsf{no\text{-}op}$. Otherwise, observe that by construction of the strand space Σ, there exist $j_1, j_2, \ldots$ such that $\cup_i E^i_{a,j_i} \subseteq \mathit{hist}_a^m(C)$, and $\mathit{hist}_a^{m+1}(C) - \mathit{hist}_a^m(C)$ is either e^i_{a,j_i+1} for some i, in which case, we take $\mathbf{a}_a = \mathsf{send}(u)$ if $e^i_{a,j_i+1} = \mathsf{sent}(u)$, and $\mathbf{a}_a = \mathsf{no\text{-}op}$ otherwise; if $\mathit{hist}_a^{m+1}(C) - \mathit{hist}_a^m(C) = \{\mathsf{recv}(u)\}$ for some u, we take $\mathbf{a}_a = \mathsf{no\text{-}op}$. Finally, take $\mathbf{a} = \langle \mathbf{a}_a : a \in \mathcal{A}\rangle$. It is not hard to check that this joint action has the desired property. For example, if $\mathit{hist}_a^{m+1}(C) - \mathit{hist}_a^m(C) = \{\mathsf{sent}(u)\}$, then $\mathbf{a}_a = \mathsf{send}(u)$. If $\mathit{hist}_a^{m+1}(C) - \mathit{hist}_a^m(C) = \{\mathsf{recv}(u)\}$, then $\mathbf{a}_a = \mathsf{no\text{-}op}$, and by the bundle properties, there must have been a corresponding send appearing in $\mathit{hist}_b^{m+1}(C)$ for some other b. In both cases, using strand transition function τ_P gives us the right result. Hence $r^C \in \mathcal{R}(P, \tau_P, I_0)$ for $C \in \mathit{Chains}(\Sigma, \mathcal{A}, A)$.

Second, consider a run $r \in \mathcal{R}(P, \tau_P, I_0)$. We need to construct a chain $C \in \mathit{Chains}(\Sigma, \mathcal{A}, A)$ such that $r = r^C$. However, for the same reasons as in the proof of Theorem 3.1, we cannot simply define the chain inductively. We use the same construction as in the proof of Theorem 3.1, which tells us how to construct a chain; essentially, the chain is constructed by "picking" the right strands from bundles. However, to apply the construction, we need to verify a few facts. From Theorem 6.1, we know that $\mathcal{R}(P, \tau_P, I_0)$ is a strand system, generated by $\langle V_a : a \in \mathcal{A}\rangle$, where $V_a = \{r_a(m) : a \in \mathcal{A}, r \in \mathcal{R}(P, \tau_P, I_0)\}$. The construction in the proof of Theorem 3.1 relied on the fact that to every history $h \in V_a$, there was a bundle in Σ for which every event associated with a strand of agent a corresponded to an event in h. The same holds in our setting: observe that $h \in V_a$ iff there exist $j_1, j_2, \ldots$ such that $\cup_i E^i_{a,j_i} \subseteq h$ and $h - \cup_i E^i_{a,j_i}$ is made up exclusively of recv events, by definition of τ_P and the fact that P is decomposable into monotone protocols. For any such history h there is a bundle B with the following strand prefixes corresponding to agent a: for each i, if $j_i = 0$, then no node of $s_a^{i,n}$ is in B (for any n), while if $j_i = k$, then the first k nodes of $s_a^{i,n}$ (for any $n \geq k$) are in B; the events in h unaccounted for by these strand prefixes are recv events, for which corresponding strands of the form s_a^u are in B. (Whatever other strands are in B are unimportant, so simply take the downward closure of the given strand prefixes.) Therefore, we can apply the construction in the proof of Theorem 3.1 to get a chain $C = B_0 \mapsto B_1 \mapsto \ldots$ with the property that $r_a(m) = \mathit{hist}_a^m(C)$. Hence, $r = r^C$. In other words, $r \in T_A(\Sigma, \mathcal{A}, A)$. $\square$

ACKNOWLEDGMENTS

We would like to thank Andre Scedrov for pointing us to fair exchange protocols as a likely source of knowledge-based specifications in security protocols. Vicky Weissman and Kevin O'Neill read a draft of this paper and provided numerous

helpful suggestions.

REFERENCES

Alpern, B. and Schneider, F. B. 1985. Defining liveness. *Information Processing Letters 21*, 181–185.

Asokan, N., Shoup, V., and Waidner, M. 1998. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*. IEEE Computer Society Press, 86–99.

Ben-Or, M., Goldreich, O., Micali, S., and Rivest, R. L. 1990. A fair protocol for signing contracts. *IEEE Transactions on Information Theory 36,* 1, 40–46.

Burrows, M., Abadi, M., and Needham, R. M. 1990. A logic of authentication. *ACM Transactions on Computer Systems 8,* 1, 18–36.

Busi, N. and Gorrieri, R. 1994. Distributed conflicts in communicating systems. In *ECOOP Workshop*. Lecture Notes in Computer Science, vol. 924. Springer, 49–65.

Cervesato, I., Durgin, N., Mitchell, J., Lincoln, P., and Scedrov, A. 2000. Relating strands and multiset rewriting for security protocol analysis. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 35–51.

Crazzolara, F. and Winskel, G. 2001. Events in security protocols. In *Proceedings of the Eighth ACM Conference on Computer and Communications Security*. ACM Press, 96–105.

Fagin, R., Halpern, J. Y., Moses, Y., and Vardi, M. Y. 1995. *Reasoning about Knowledge*. The MIT Press.

Gray III, J. W. and Syverson, P. F. 1998. A logical approach to multilevel security of probabilistic systems. *Distributed Computing 11,* 2, 73–90.

Grove, A. J. 1995. Naming and identity in epistemic logic II: a first-order logic for naming. *Artificial Intelligence 74,* 2, 311–350.

Grove, A. J. and Halpern, J. Y. 1993. Naming and identity in epistemic logics, Part I: the propositional case. *Journal of Logic and Computation 3,* 4, 345–378.

Halpern, J. Y. 2000. A note on knowledge-based programs and specifications. *Distributed Computing 13*, 145–153.

Halpern, J. Y., Moses, Y., and Tuttle, M. R. 1988. A knowledge-based analysis of zero knowledge. In *Proc. 20th ACM Symp. on Theory of Computing*. 132–147.

McLean, J. 1994. Security models. In *Encyclopedia of Software Engineering*, J. Marciniak, Ed. Wiley Press.

Palamidessi, C. 1997. Comparing the expressive power of the synchronous and the asynchronous pi-calculus. In *Conference Record of the Twenty-Fourth Annual ACM Symposium on Principles of Programming Languages*. ACM Press, 256–265.

Shmatikov, V. and Mitchell, J. C. 2000. Analysis of a fair exchange protocol. In *Seventh Annual Symposium on Network and Distributed System Security (NDSS 2000)*. 119–128.

Snekkenes, E. 1992. Roles in cryptographic protocols. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 105–119.

Stubblebine, S. and Wright, R. 1996. An authentication logic supporting synchronization, revocation, and recency. In *3rd ACM Conference on Computer and Communications Security*. ACM Press.

Syverson, P. 1990. A logic for the analysis of cryptographic protocols. NRL Report 9305, Naval Research Laboratory.

Syverson, P. 1999. Towards a strand semantics for authentication logic. *Electronic Notes in Theoretical Computer Science 20*.

Thayer, F. J., Herzog, J. C., and Guttman, J. D. 1999a. Mixed strand spaces. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press.

Thayer, F. J., Herzog, J. C., and Guttman, J. D. 1999b. Strand spaces: Proving security protocols correct. *Journal of Computer Security 7,* 2/3, 191–230.

???