Strand Spaces With Explicit State-Passing

John D. Ramsdell

June 18, 2014

The introduction has not been written yet. Some observations follow.

This paper adds state to strand spaces. In strand spaces, a strand is a sequence of message transmission and reception events. An execution in strand spaces is a set of strands along with a pairing of transmission and reception events such that the same message is passed between the two events.

The addition of state-passing involves adding a state synchronization event associated with a transition in a state machine. An execution in strand spaces with transition events augments message-passing with state-passing. State synchronization events are paired such that the post-state in one event is the same as the pre-state in the next event.

This approach to adding state to strand spaces preserves an important property of strand spaces. One can always trace a path backwards from the use of a message to some location at which it originated. This fact has been verified using PVS using the methods described in [3]. More generally, all of the theorems and lemmas in this paper have proved in PVS with the exception of Lemmas 7 and 8 in Section 4.

There are holes in this paper marked by $***$. Any help filling them would be greatly appreciated.

1 Strand Spaces With Transitions

The parameters to the strand space theory with transitions are

1. a set of messages ($\mathcal{M}$),
2. a carried by relation ($\sqsubseteq \subseteq \mathcal{M} \times \mathcal{M}$),
3. a set of states ($\mathcal{Q}$),

4. a set of initial states ($\mathcal{I} \subseteq \mathcal{Q}$),

5. a state transition relation ($\mathcal{T} \subseteq \mathcal{Q} \times \mathcal{Q}$), and

6. a state encoding function ($enc \in \mathcal{Q} \to \mathcal{M}$).

The set of messages $\mathcal{M}$ is often the carrier set of a message algebra. Intuitively, a message m_0 is carried by m_1 ($m_0 \sqsubseteq m_1$) if it is possible to extract m_0 from m_1. An infinite sequence of states π is a *path* iff $\forall i \in \mathbb{N}.\, (\pi(i), \pi(i{+}1)) \in \mathcal{T}$ with $\pi(0) \in \mathcal{I}$.

In strand space theory, the *trace* of a strand is a linearly ordered sequence of events $e_0 \Rightarrow \cdots \Rightarrow e_{n-1}$, and an *event* is a message transmission $+m$ or a reception $-m$, where $m \in \mathcal{M}$, or a state synchronization $\circ t$, where $t \in \mathcal{T}$. A *strand space* $\mathcal{S}$ is a map from a set of strands to a set of traces. We choose the set of strands to be a prefix of the natural numbers, so a strand space is finite sequence of traces. The set of strands of strand space $\mathcal{S}$ is $\mathcal{Z}(\mathcal{S}) = Dom(\mathcal{S})$.

A node names an event in a strand space. The set of *nodes* of strand space $\mathcal{S}$ is $\{(z, i) \mid z \in \mathcal{Z}(\mathcal{S}), 0 \leq i < |\mathcal{S}(z)|\}$, and the event at a node is $evt_{\mathcal{S}}(z, i) = \mathcal{S}(z)(i)$. A node is a *transition node* in $\mathcal{S}$ iff the event at the node is a state synchronization. The set of nodes of $\mathcal{S}$ is $\mathcal{N}(\mathcal{S})$ and the set of transition nodes of $\mathcal{S}$ is $\mathcal{N}^{\circ}(\mathcal{S})$.

A state q carries a message iff its encoding $enc(q)$ carries it. A message *appears* in trace c at index i iff

1. $c(i)$ is the transmission $+m$, it is carried by m, and it is not carried by any event earlier in the trace, or

2. $c(i)$ is the synchronization event $\circ(q_0, q_1)$, the message is carried by q_1, and it is not carried by q_0 or any event earlier in the trace.

The model of execution is a bundle. The triple $\mathcal{B} = (\mathcal{S}, \to, \rightsquigarrow)$ is a *bundle* iff it defines a finite directed acyclic graph, where the vertices are the nodes of $\mathcal{S}$, and an edge represents communication ($\to$), state passing ($\rightsquigarrow$), or strand succession ($\Rightarrow$) in $\mathcal{S}$.

For communication, if $n_0 \to n_1$, then there is a message t such that $evt_{\mathcal{S}}(n_0) = +m$ and $evt_{\mathcal{S}}(n_1) = -m$. For each reception node n_1, there is a unique transmission node n_0 with $n_0 \to n_1$.

For state passing, if $n_0 \rightsquigarrow n_1$, then n_0 and n_1 are transition nodes, and there is a state q such that $evt_{\mathcal{S}}(n_0) = \circ(q_0, q)$ and $evt_{\mathcal{S}}(n_1) = \circ(q, q_1)$. Node n_1 is a *start node* iff it is a transition node and there does not exists

a node n_0, such that $n_0 \rightsquigarrow n_1$. If n is a start node with $evt_{\mathcal{S}}(n) = \circ(q_0, q_1)$ then $q_0 \in \mathcal{I}$. Finally, there exists an $\ell \in \mathbb{N}$ and bijective $f \in \mathcal{N}^{\circ}(\mathcal{S}) \to \mathbb{Z}_{\ell}$ such that

$$\forall n_0, n_1 \in \mathcal{N}^{\circ}(\mathcal{S}).\, n_0 \rightsquigarrow n_1 \text{ iff } f(n_0) + 1 = f(n_1), \tag{1}$$

and some path π such that

$$\forall n \in \mathcal{N}^{\circ}(\mathcal{S}).\, evt_{\mathcal{S}}(n) = \circ(\pi(f(n)), \pi(f(n) + 1)). \tag{2}$$

Equation 1 ensures the transition nodes are totally ordered. Equation 2 is unnecessary when considering finite paths through state space, however the use of finite paths complicates proofs in PVS so they will be excluded here. For a bundle $\mathcal{B}$, its associated strand space will be denoted $\mathcal{S}_{\mathcal{B}}$ unless the association is clear from the context.

Each acyclic graph has a transitive irreflexive relation $\prec$ on its vertices. The relation specifies the causal ordering of nodes in a bundle. A transitive irreflexive binary relation is also called a strict order. In a bundle, when $\prec$ is restricted to transition nodes, it is can be shown that it is identical to $\rightsquigarrow^+$, where $\rightsquigarrow^+$ is the transitive closure of $\rightsquigarrow$. Furthermore, Equation 1 can be used to derive the following relation between node ordering and the function f.

$$\forall n_0, n_1 \in \mathcal{N}^{\circ}(\mathcal{S}).\, n_0 \prec n_1 \text{ iff } f(n_0) < f(n_1) \tag{3}$$

A message t *originates* in bundle $\mathcal{B} = (\mathcal{S}, \rightarrow, \rightsquigarrow)$ at node n iff

1. it appears in $\mathcal{S}(z)$ at index i and $n = (z, i)$, or
2. n is a start node, $evt_{\mathcal{S}}(n) = \circ(q_0, q_1)$, and m is carried by q_0.

A message m is *non-originating* in bundle $\mathcal{B}$, written $non(\mathcal{B}, m)$, if it originates at no node. A message m *uniquely originates* in bundle $\mathcal{B}$ at node n, written $uniq(\mathcal{B}, m, n)$, if it originates at n and nowhere else. With the definitions of origination and bundles given here, strand spaces with transitions retains a key property of the original version of strand spaces.

Lemma 1. *If message m is carried by $evt_{\mathcal{S}_{\mathcal{B}}}(n)$, then m originates in $\mathcal{B}$.*

Proof. By induction on the graph of $\mathcal{B}$ and a case analysis of the events at n.

1. If n is a transmission, then either m originates at n or m is carried earlier in the strand by the definition of appears.

2. If n is a reception, then there is an earlier transmission that carries m.

3. If $evt_{\mathcal{S}_\mathcal{B}}(n) = \circ(q_0, q_1)$ and $m \sqsubseteq q_0$, then n is a start node or for some n', $n' \rightsquigarrow n$, $evt_{\mathcal{S}_\mathcal{B}}(n') = \circ(q'_0, q'_1)$, and $m \sqsubseteq q'_1$.

4. If $evt_{\mathcal{S}_\mathcal{B}}(n) = \circ(q_0, q_1)$ and $m \sqsubseteq q_1$, then $m \sqsubseteq q_0$, or m appears at n, or m is carried earlier in the strand. □

In the remainder of this section, the theory of strand spaces used in the proofs has been simplified. In the full theory, origination assumptions can be inherited from roles. See [3] for all the gory details.

When a bundle is a run of a protocol, the behavior of each strand is constrained by a role. Adversarial strands are constrained by roles as are non-adversarial strands. A *protocol* is a set of roles, and a *role* is a set of traces. A trace c is an *instance* of role r iff c is a prefix of some member of r. For protocol P, bundle $\mathcal{B} = (\mathcal{S}, \rightarrow, \rightsquigarrow)$ is a *run of protocol* P iff there exists a role assignment $ra \in \mathcal{Z}(\mathcal{S}) \rightarrow P$ such that for all $z \in \mathcal{Z}(\mathcal{S})$, $\mathcal{S}(z)$ is an instance of $ra(z)$.

1.1 Skeletons

In this paper, a skeleton will be specified using a sentence in first-order logic. The atomic formula $\mathsf{htin}(z, h, c)$ asserts that strand z has a length of at least h, and its trace is a prefix of trace c. The formula $n_0 \ll n_1$ asserts node n_0 precedes node n_1. The formula $\mathsf{non}(m)$ asserts that message m is non-originating, and $\mathsf{uniq}(m, n)$ asserts that message m uniquely originates at node n. Finally, the formula $\mathsf{sends}(n, m)$ asserts that the event at node n is a transmission of message m. The roles of the protocol serve as function symbols.

For bundle $\mathcal{B}$, variable assignment α, and formula Φ, satisfaction $\mathcal{B}, \alpha \models \Phi$ is defined using the semantics for atomic formulas specified in Figure 1. A bundle $\mathcal{B}$ is described by a skeleton iff the skeleton's sentence Φ is modeled by $\mathcal{B}$, written $\mathcal{B} \models \Phi$.

2 Message Model

Typically, messages are modeled by elements of an order-sorted algebra [1]. An order-sorted algebra is a generalization of a many-sorted algebra in which

$$
\begin{array}{ll}
\mathcal{B}, \alpha \models x = y & \text{iff } \alpha(x) = \alpha(y); \\
\mathcal{B}, \alpha \models \mathsf{htin}(z, h, c) & \text{iff } |\mathcal{B}_{\mathcal{S}}(\alpha(z))| \geq \alpha(h) \text{ and} \\
 & \quad \mathcal{B}_{\mathcal{S}}(\alpha(z)) \text{ is a prefix of } \alpha(c); \\
\mathcal{B}, \alpha \models n_0 \ll n_1 & \text{iff } \alpha(n_0) \prec_{\mathcal{B}} \alpha(n_1); \\
\mathcal{B}, \alpha \models \mathsf{non}(m) & \text{iff } non(\mathcal{B}, \alpha(m)); \\
\mathcal{B}, \alpha \models \mathsf{uniq}(m, n) & \text{iff } uniq(\mathcal{B}, \alpha(m), \alpha(n)); \\
\mathcal{B}, \alpha \models \mathsf{sends}(n, m) & \text{iff } evt_{\mathcal{B}_{\mathcal{S}}}(\alpha(n)) = +\alpha(m).
\end{array}
$$

Figure 1: Satisfaction

sorts may be partially ordered. The carrier sets associated with ordered sorts are related by the subset relation.

Figure 2 shows the signature of the algebra used in examples in this paper. Sort M is the sort of all messages. Messages of sort A (asymmetric keys), sort S (symmetric keys), sort D (data), and sort E (text) are called *atoms*. Messages are atoms, tag constants, or constructed using encryption $\{\!|\cdot|\!\}_{(\cdot)}$, hashing $\#(\cdot)$, and pairing $(\cdot, \cdot)$, where the comma operation is right associative and parentheses are omitted when the context permits.

The algebra $\mathbb{A}$ is the initial quotient term algebra over the signature. The canonical representative for each element in the algebra is the term that contains no occurrences of the inverse operation $(\cdot)^{-1}$. At times, we conflate a message with its canonical representative. The carrier set $\mathbb{A}_{\mathsf{M}}$ for sort M is what is used to instantiate strand spaces with transitions. For sort S in the signature, we write $m : S$ for $m \in \mathbb{A}_S$. For skeleton formulas, $\mathcal{B}, \alpha \models m : S$ iff $\alpha(m) \in \mathbb{A}_S$. For strands and nodes, sorts Z and N have been added with the property that $\mathcal{B}, \alpha \models z{:}\mathsf{Z}$ iff $\alpha(z) \in \mathcal{Z}(\mathcal{S}_{\mathcal{B}})$ and $\mathcal{B}, \alpha \models n{:}\mathsf{N}$ iff $\alpha(n) \in \mathcal{N}(\mathcal{S}_{\mathcal{B}})$.

A message m_0 is *carried by* m_1, written $m_0 \sqsubseteq m_1$ iff m_0 can be extracted from a reception of m_1, assuming plaintext is extractable from encryptions. In other words, $\sqsubseteq$ is the smallest reflexive, transitive relation such that $m_0 \sqsubseteq m_0$, $m_0 \sqsubseteq (m_0, m_1)$, $m_1 \sqsubseteq (m_0, m_1)$, and $m_0 \sqsubseteq \{\!|m_0|\!\}_{m_1}$.

The roles that constrain adversarial behavior are defined by the functions in Figure 3. The role defined by the function is all the traces that it generates. For example, the role associated with the function *pair* is $\{pair(m_0, m_1) \mid m_0, m_1 : \mathsf{M}\}$. For the encryption related roles, $k : \mathsf{A}|\mathsf{S}$ asserts that k is either a symmetric or asymmetric key. For the create role, $m : \mathsf{A}|\mathsf{S}|\mathsf{D}|\mathsf{E}$ asserts that m is an atom.

Sorts:	M, A, S, D, E		
Subsorts:	$\mathsf{A} < \mathsf{M}$, $\mathsf{S} < \mathsf{M}$, $\mathsf{D} < \mathsf{M}$, $\mathsf{E} < \mathsf{M}$		
Operations:	$(\cdot,\cdot)$	$: \mathsf{M} \times \mathsf{M} \to \mathsf{M}$	Pairing
	$\{\!\lfloor \cdot \rfloor\!\}_{(\cdot)}$	$: \mathsf{M} \times \mathsf{A} \to \mathsf{M}$	Asymmetric encryption
	$\{\!\lfloor \cdot \rfloor\!\}_{(\cdot)}$	$: \mathsf{M} \times \mathsf{S} \to \mathsf{M}$	Symmetric encryption
	$\#$	$: \mathsf{M} \to \mathsf{M}$	Hashing
	$(\cdot)^{-1}$	$: \mathsf{A} \to \mathsf{A}$	Asymmetric key inverse
	$(\cdot)^{-1}$	$: \mathsf{S} \to \mathsf{S}$	Symmetric key inverse
	$\mathsf{a}_i, \mathsf{b}_i$	$: \mathsf{A}$	Asymmetric key constants
	s_i	$: \mathsf{S}$	Symmetric key constants
	d_i	$: \mathsf{D}$	Data constants
	e_i	$: \mathsf{E}$	Text constants
	g_i	$: \mathsf{M}$	Tag constants
Equations:	${\mathsf{a}_i}^{-1} = \mathsf{b}_i \quad {\mathsf{b}_i}^{-1} = \mathsf{a}_i$		$(i \in \mathbb{N})$
	$\forall k : \mathsf{A}.\, (k^{-1})^{-1} = k$		$\forall k : \mathsf{S}.\, k^{-1} = k$

Figure 2: Crypto Algebra Signature

$$
\begin{aligned}
\mathit{create}(m : \mathsf{A}|\mathsf{S}|\mathsf{D}|\mathsf{E}) &= +m \qquad \mathit{tag}_i = +\,\mathsf{g}_i \\
\mathit{pair}(m_0 : \mathsf{M}, m_1 : \mathsf{M}) &= -m_0 \Rightarrow -m_1 \Rightarrow +(m_0, m_1) \\
\mathit{sep}(m_0 : \mathsf{M}, m_1 : \mathsf{M}) &= -(m_0, m_1) \Rightarrow +m_0 \Rightarrow +m_1 \\
\mathit{enc}(m : \mathsf{M}, k : \mathsf{A}|\mathsf{S}) &= -m \Rightarrow -k \Rightarrow +\{\!\lfloor m \rfloor\!\}_k \\
\mathit{dec}(m : \mathsf{M}, k : \mathsf{A}|\mathsf{S}) &= -\{\!\lfloor m \rfloor\!\}_k \Rightarrow -k^{-1} \Rightarrow +m \\
\mathit{hash}(m : \mathsf{M}) &= -m \Rightarrow +\#m
\end{aligned}
$$

Figure 3: Adversary Traces

3 TPM State Model

The model of a Trusted Platform Module (TPM) is very simple. It models a TPM as a single Platform Configuration Register (PCR) that can be extended and reset. TPM state is modeled by augmenting the algebra signature with a sort Q for the states, and two additional operations.

$$\begin{array}{ll} \mathsf{bt} : \mathsf{Q} & \text{Boot state} \\ \mathsf{ex} : \mathsf{M} \times \mathsf{Q} \to \mathsf{Q} & \text{Extend state} \end{array}$$

The set of states in this model is $\mathcal{Q} = \mathbb{A}_{\mathsf{Q}}$.

A TPM state is encoded as a message that represents the PCR associated with the state.

$$\begin{aligned} enc &\in \mathcal{Q} \to \mathcal{M} \\ enc(\mathsf{bt}) &= \mathsf{g}_0 \\ enc(\mathsf{ex}(m, q)) &= \#(m, enc(q)) \end{aligned}$$

The PCR for the boot state is known to the adversary.

The TPM transition relation is $\mathcal{T} \subseteq \mathcal{Q} \times \mathcal{Q}$, where

$$\begin{array}{lllll} (q_0, q_1) \in \mathcal{T} & \text{iff} & q_1 = \mathsf{bt} & & \text{(boot)} \\ & \text{or} & \exists m \in \mathcal{M}.\, q_1 = \mathsf{ex}(m, q_0) & & \text{(extend)} \\ & \text{or} & q_0 = q_1 & & \text{(observe)} \end{array}$$

Observation transitions are used with the TPM quote operation. The set of TPM initial states $\mathcal{I} \subseteq \mathcal{Q}$ is the singleton $\{\mathsf{bt}\}$.

Useful properties about paths through state space are given by the Init Extend and the Prefix Extend Lemmas.

Lemma 2 (Init Extend)**.**

$$\begin{aligned} &\forall \pi, m : \mathsf{M}, q : \mathsf{Q}, k \in \mathbb{N}. \\ &\quad \pi(k) = \mathsf{ex}(m, q) \\ &\qquad \supset \exists j \in \mathbb{N}.\, j < k \\ &\qquad\quad \wedge \pi(j) = q \wedge \pi(j+1) = \mathsf{ex}(m, q) \end{aligned}$$

Lemma 3 (Prefix Extend)**.**

$$\begin{aligned} &\forall \pi, m : \mathsf{M}, q : \mathsf{Q}, i, k \in \mathbb{N}.\, i \leq k \\ &\quad \wedge \pi(k) = \mathsf{ex}(m, q) \\ &\qquad \supset \pi(i) \text{ is a subterm of } \pi(k) \\ &\qquad\quad \vee \exists j \in \mathbb{N}.\, i \leq j < k \\ &\qquad\qquad \wedge \pi(j) = q \wedge \pi(j+1) = \mathsf{ex}(m, q) \end{aligned}$$

These two lemmas are used to prove the one employed to prove the security goal in the envelope protocol.

Lemma 4 (State Split)**.**

$$\begin{array}{l}\forall \pi, m, m_0, m_1 : \mathsf{M}, q : \mathsf{Q}, i, k \in \mathbb{N}.\, i \leq k \\ \quad \wedge\, \pi(i) = \mathsf{ex}(m_0, \mathsf{ex}(m, q)) \\ \quad \wedge\, m_0 \neq m_1 \wedge \pi(k) = \mathsf{ex}(m_1, \mathsf{ex}(m, q)) \\ \quad\quad \supset \exists j_0, j_1 \in \mathbb{N}.\, j_0 < i < j_1 < k \\ \quad\quad\quad \wedge\, \pi(j_0) = \pi(j_1) = q \\ \quad\quad\quad \wedge\, \pi(j_0 + 1) = \pi(j_1 + 1) = \mathsf{ex}(m, q)\end{array}$$

3.1 A Bridge Lemma

A *bridge* lemma imports a lemma from the state model into strand spaces with transitions. The bridge lemma for Lemma 2 and its proof follow.

Lemma 5 (Init Extend Bridge)**.**

$$\begin{array}{l} n \in \mathcal{N}^{\circ}(\mathcal{B}) \wedge evt_{\mathcal{S}}(n) = \circ(\mathsf{ex}(m, q_0), q_1) \\ \quad \supset \exists n'.\, n' \prec n \wedge evt_{\mathcal{S}}(n') = \circ(q_0, \mathsf{ex}(m, q_0))\end{array}$$

Proof. Because $\mathcal{B}$ is a bundle, Equations 2 and 3 apply. Instantiating Equation 2 with n adds the assumption

$$evt_{\mathcal{S}}(n) = \circ(\pi(f(n)), \pi(f(n) + 1)),$$

so $\pi(f(n)) = \mathsf{ex}(m, q_0)$. Instantiating Lemma 2 adds the assumption

$$\exists j \in \mathbb{N}.\, j < f(n) \wedge \pi(j) = q_0 \wedge \pi(j + 1) = \mathsf{ex}(m, q_0).$$

Since an existential in an assumption is equivalent to a universal over an implication, we specialize to add three assumptions,

$$j < f(n) \wedge \pi(j) = q_0 \wedge \pi(j + 1) = \mathsf{ex}(m, q_0).$$

Function f is bijective and therefore surjective. There exists an n' such that $f(n') = j$. Eliminating j gives

$$f(n') < f(n) \wedge \pi(f(n')) = q_0 \wedge \pi(f(n') + 1) = \mathsf{ex}(m, q_0).$$

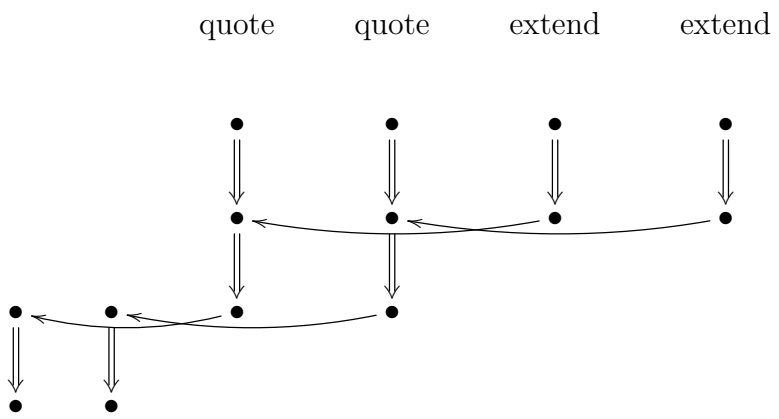

Figure 4: Result Skeleton

Instantiating Equation 3 with n' and n gives $n' \prec n$. Note that this instantiation makes use of the backward implication of Equation 3. Instantiating Equation 2 with n' adds

$$evt_{\mathcal{S}}(n') = \circ(\pi(f(n')), \pi(f(n') + 1)),$$

so $evt_{\mathcal{S}}(n') = \circ(q_0, \mathsf{ex}(m, q_0))$. Instantiating the conclusion of the lemma with n' completes the proof. □

3.2 State Split Protocol Analysis

The State Split Protocol is a greatly simplified version of the Envelope Protocol. An analysis of the Envelope Protocol is described in [4]. The State Split Protocol shares interesting analysis issues.

The protocol uses tag $quo = \mathsf{g}_1$ for "quote", $obt = \mathsf{g}_2$ for "obtain", and $ref = \mathsf{g}_3$ for "refuse". The traces of the non-adversarial roles are

$$\begin{array}{ll} boot(q : \mathsf{Q}) & = \circ(q, \mathsf{bt}) \\ extend(q : \mathsf{Q}, m : \mathsf{M}) & = -m \Rightarrow \circ(q, \mathsf{ex}(m, q)) \\ quote(q : \mathsf{Q}, t : \mathsf{E}, k : \mathsf{A}) & = -t \Rightarrow \circ(q, q) \Rightarrow +mkq(t, q, k) \end{array}$$

where the make quote function is $mkq(t, q, k) = \{|\#(quo, enc(q), t)|\}_k$. In this section, we consider only bundles that are runs of this protocol.

$$
\begin{array}{l}
\forall n_0, n_1 : \mathsf{N}, q : \mathsf{Q}, t_0, t_1 : \mathsf{E}, k : \mathsf{A}. \\
\quad \mathsf{send}(n_0, mkq(t_0, \mathsf{ex}(obt, q), k)) \wedge \mathsf{non}(k) \\
\qquad \wedge \mathsf{send}(n_1, mkq(t_1, \mathsf{ex}(ref, q), k)) \\
\qquad \supset \exists z_0, z_1, z_2, z_3 : \mathsf{Z}. \\
\qquad\quad n_0 = (z_0, 2) \wedge n_1 = (z_1, 2) \\
\qquad\qquad \wedge \mathsf{htin}(z_0, 3, quote(q, obt, k)) \\
\qquad\qquad \wedge \mathsf{htin}(z_1, 3, quote(q, ref, k)) \\
\qquad\qquad \wedge \mathsf{htin}(z_2, 2, extend(q, obt)) \\
\qquad\qquad \wedge \mathsf{htin}(z_3, 2, extend(q, ref)) \\
\qquad\qquad \wedge (z_2, 1) \ll (z_0, 1) \wedge (z_3, 1) \ll (z_1, 1) \\
\qquad\qquad \wedge ((z_2, 1) \ll (z_3, 1) \vee (z_3, 1) \ll (z_2, 1))
\end{array}
$$

Figure 5: Shape Analysis Sentence

The point-of-view skeleton is described by

$$
\begin{array}{l}
\exists n_0, n_1 : \mathsf{N}, q : \mathsf{Q}, t_0, t_1 : \mathsf{E}, k : \mathsf{A}. \\
\quad \mathsf{send}(n_0, mkq(t_0, \mathsf{ex}(obt, q), k)) \wedge \mathsf{non}(k) \\
\qquad \wedge \mathsf{send}(n_1, mkq(t_1, \mathsf{ex}(ref, q), k))
\end{array}
$$

A diagram of the result skeleton is in Figure 4, and its associated shape analysis sentence is in Figure 5. Note that in all bundles of the result skeleton, the transition nodes are totally ordered, but the transition nodes in the result skeleton are not. Thus the result skeleton is not realized.

Let's compare a state threading analysis of the State Split Protocol with a hypothetical analysis using strand spaces with transitions. The first point to make is that bridge lemmas are simpler and more obvious because of the fact that a state synchronization event is a transition, in contrast with a node annotation which is a set of transitions. For this problem, Lemma 5 is the relevant bridge lemma. This bridge lemma is used to derive Lemma 6, which is satisfied only in bundles that are runs of the protocol.

Lemma 6 (Quote Extend Implies Extend)**.**

$$
\begin{array}{l}
\forall z : \mathsf{Z}, q : \mathsf{Q}, m : \mathsf{M}, t : \mathsf{E}, k : \mathsf{A}. \\
\quad \mathsf{htin}(z, 3, quote(\mathsf{ex}(m, q), t, k)) \\
\qquad \supset \exists z_0 : \mathsf{Z}.\, \mathsf{htin}(z_0, 2, extend(q, m)) \wedge (z_0, 1) \ll (z, 1)
\end{array}
$$

For this problem, the quote strands are inferred using ordinary authentication tests. The extend strands are inferred using the lemma above. The

remaining node orderings are derived using the fact that transition nodes are totally ordered.

The strand spaces with transitions approach follows closely the state threading approach used in [4]. In the state threading approach, authentication tests occur at nodes carrying a representation of the state. In the strand spaces with transitions approach, those tests occur at transition nodes. In particular, in this problem, after inferring the existence of the quote strands, the question is how to explain the first state in the second node of a quote strand. The answer is to augment with an extend strand.

4 The $\mathcal{B}$, $\mathcal{C}$, ϕ Model

Strand spaces with transitions is a natural way of adding state to strand spaces. To bundles that contain message-passing edges, it adds state-passing edges, and the rest follows. However, the state-passing model has a serious shortcoming. State and message-passing are intertwined in a way that makes it hard to reuse results on slightly different problems.

The $\mathcal{B}$, $\mathcal{C}$, ϕ model [2] was designed to address this shortcoming. In this model, states are related using a labeled transition system. The labels $\mathcal{L}$ are messages ($\mathcal{L} \subseteq \mathcal{M}$), and the transition system $\overline{\mathcal{T}}$ is in $\mathcal{Q} \times \mathcal{L} \times \mathcal{Q}$. A computation $\mathcal{C} = (\lambda, \pi)$ is a pair where λ is an infinite sequence of labels, π is a path, and $\forall i \in \mathbb{N}.\, (\pi(i), \lambda(i), \pi(i+1)) \in \overline{\mathcal{T}}$. In general, variables related to ones defined by strand spaces with transitions will be barred.

In the $\mathcal{B}$, $\mathcal{C}$, ϕ model, a state synchronization event is a label, not a transition. A node is a *neutral node* in $\overline{\mathcal{S}}$ iff the event at the node is a state synchronization. The set of neutral nodes of $\overline{\mathcal{S}}$ is $\bar{\mathcal{N}}^{\circ}(\overline{\mathcal{S}})$. The bundle $\overline{\mathcal{B}} = (\overline{\mathcal{S}}, \rightarrow)$ omits the state-passing edges $\rightsquigarrow$ from its associated graph along with the constraints associated with state-passing. In short, a bundle $\overline{\mathcal{B}}$ is just a strand space bundle augmented with neutral nodes.

Let $\mathcal{C} = (\lambda, \pi)$ and ℓ be the number of neutral nodes in $\overline{\mathcal{B}}$. We define $\overline{\mathcal{B}}$, $\mathcal{C}$, ϕ to be a *compatible triple* iff $\phi \in \bar{\mathcal{N}}^{\circ}(\overline{\mathcal{S}}) \rightarrow \mathbb{Z}_{\ell}$ and is bijective,

$$\forall n_0, n_1 \in \bar{\mathcal{N}}^{\circ}(\overline{\mathcal{S}}).\, n_0 \prec n_1 \supset \phi(n_0) < \phi(n_1), \tag{4}$$

and

$$\forall n \in \bar{\mathcal{N}}^{\circ}(\overline{\mathcal{S}}).\, evt_{\overline{\mathcal{S}}}(n) = \circ\lambda(\phi(n)). \tag{5}$$

A bundle $\overline{\mathcal{B}}$ is *compatible with* $\mathcal{C}$ iff there exists a ϕ such that $\overline{\mathcal{B}}$, $\mathcal{C}$, ϕ is a

compatible triple. Bundle $\overline{\mathcal{B}}$ is a *compatible bundle* iff it is compatible with some computation $\mathcal{C}$. All of these definitions are from [2].

We investigate the relationship between strand spaces with labels and strand spaces with transitions while seeking to preserve Lemma 1. As a first pass, we assume the message algebra has a pairing operation. Let $\mathcal{R} = \mathit{Ran}(\mathit{enc})$ be the range of *enc*, and $\mathcal{L} = \{(m_0, m_1) \mid m_0, m_1 \in \mathcal{R}\}$ so that a label encodes a transition. With this choice of labels, the definition of appears and origination in strand spaces with labels is obvious, and Lemma 1 applies.

Let $\Lambda(\mathcal{B})$ be the strand spaces with labels bundle that results from deleting all state-passing edges $\rightsquigarrow$, and replacing state synchronization events of the form $\circ(q_0, q_1)$ with $\circ(\mathit{enc}(q_0), \mathit{enc}(q_1))$.

Lemma 7 (Faithful Labels)**.** *If $\mathcal{B}$ is a bundle, then $\Lambda(\mathcal{B})$ is a compatible bundle.*

Proof. By the properties of a bundle, there exists a bijection f where $f \in \mathcal{N}^{\circ}(\overline{\mathcal{S}}) \to \mathbb{Z}_{\ell}$. By Equation 2, there is a path π that can be extended into a computation $\mathcal{C}$ using the chosen labeling. $\Lambda(\mathcal{B})$, $\mathcal{C}$, f are a compatible triple. □

Notice that Equation 3 is a biconditional but Equation 4 is an implication. The addition of the state-passing edges is what enables the strengthening of the implication to a biconditional. Therefore, the proof a bridge lemma in $\mathcal{B}$, such as the one for Lemma 5, also applies to $\Lambda(\mathcal{B})$.

Assume the encoding function is injective. The PCR encoding function from Section 3 is injective. Let decoding function $\mathit{dec} \in \mathcal{R} \to \mathcal{Q}$ be the inverse of *enc*.

Let $\Delta(\overline{\mathcal{B}}, \mathcal{C}, \phi)$ be result of replacing the events in $\overline{\mathcal{B}}$ of the form $\circ(m_0, m_1)$ with $\circ(\mathit{dec}(m_0), \mathit{dec}(m_1))$, and adding edges of the form $n_0 \rightsquigarrow n_1$ when $n_0, n_1 \in \mathit{Dom}\,\phi$ and $\phi(n_0) + 1 = \phi(n_1)$.

Lemma 8 (Faithful Transitions)**.** *If enc is injective and $\overline{\mathcal{B}}$, $\mathcal{C}$, ϕ is a compatible triple, then $\Delta(\overline{\mathcal{B}}, \mathcal{C}, \phi)$ is a bundle.*

Proof. By Equation 1 on transition node orderings. □

4.1 Constraints on Labels

Of course, encoding transitions as labels defeats the whole purpose of the $\mathcal{B}$, $\mathcal{C}$, ϕ model! State and message-passing are intertwined as in the state-passing model, all be it in an encoded form.

To satisfy Lemma 1, we add two relations $\sqsubseteq_0, \sqsubseteq_1 \in \mathcal{M} \times \mathcal{L}$ with the property that for all $q_0, q_1 \in \mathcal{Q}$, $m_0 \in \mathcal{L}$, and $m \in \mathcal{M}$,

$$m \sqsubseteq_0 m_0 \text{ iff } (q_0, m_0, q_1) \in \overline{\mathcal{T}} \wedge m \sqsubseteq q_0 \tag{6}$$

and

$$m \sqsubseteq_1 m_0 \text{ iff } (q_0, m_0, q_1) \in \overline{\mathcal{T}} \wedge m \sqsubseteq q_1. \tag{7}$$

The definition of appears becomes: a message m *appears* in trace c at index i iff

1. $c(i)$ is the transmission $+m_0$, m is carried by m_0, and m is not carried by any event earlier in the trace, or
2. $c(i)$ is the synchronization event $\circ m_0$, $m \sqsubseteq_1 m_0$, $m \not\sqsubseteq_0 m_0$, and m is not carried by any event earlier in the trace,

where a message m is carried in event $\circ m_0$ iff $m \sqsubseteq_0 m_0$ or $m \sqsubseteq_1 m_0$. (Only Item 2 changed.) The definition of origination requires a similar modification.

Consider the following set of labels for the TPM model in Section 3. The labels use tag $btl = \mathsf{g}_4$ for "boot label", $exl = \mathsf{g}_5$ for "extend label", and $obl = \mathsf{g}_6$ for "observe label". The set of labels $\mathcal{L}$ is

$$\begin{array}{rcl} (btl, m_0) \in \mathcal{L} & \text{if} & m_0 \in \mathcal{R} \\ (exl, m, m_0) \in \mathcal{L} & \text{if} & m \in \mathcal{M} \wedge m_0 \in \mathcal{R} \\ (obl, m_0) \in \mathcal{L} & \text{if} & m_0 \in \mathcal{R} \end{array}$$

The TPM labeled transition relation is $\overline{\mathcal{T}} \subseteq \mathcal{Q} \times \mathcal{L} \times \mathcal{Q}$, where

$$\begin{array}{lcll} (q_0, m_0, \mathsf{bt}) \in \overline{\mathcal{T}} & \text{if} & m_0 = (btl, enc(q_0)) & \text{(boot)} \\ (q_0, m_0, \mathsf{ex}(m, q_0)) \in \overline{\mathcal{T}} & \text{if} & m_0 = (exl, m, enc(q_0)) & \text{(extend)} \\ (q_0, m_0, q_0) \in \overline{\mathcal{T}} & \text{if} & m_0 = (obl, enc(q_0)) & \text{(observe)} \end{array}$$

The relations $\sqsubseteq_0, \sqsubseteq_1 \in \mathcal{M} \times \mathcal{L}$ are

$$\begin{array}{cc} m_1 \sqsubseteq_0 (btl, m_0) = m_1 \sqsubseteq m_0 & m_1 \sqsubseteq_1 (btl, m_0) = \mathit{false} \\ m_1 \sqsubseteq_0 (exl, m, m_0) = m_1 \sqsubseteq m_0 & m_1 \sqsubseteq_1 (exl, m, m_0) = m_1 \sqsubseteq \#(m, m_0) \\ m_1 \sqsubseteq_0 (obl, m_0) = m_1 \sqsubseteq m_0 & m_1 \sqsubseteq_1 (obl, m_0) = m_1 \sqsubseteq m_0 \end{array}$$

In this instance of strand spaces with labels, Lemma 1 applies, however, have we made any real progress toward partitioning state and message-passing? The answer is no because a label still encodes a transition.

$$\begin{array}{rcl} (btl, m_0) & \Longrightarrow & (dec(m_0), \mathsf{bt}) \\ (exl, m, m_0) & \Longrightarrow & (dec(m_0), \mathsf{ex}(m, dec(m_0))) \\ (obl, m_0) & \Longrightarrow & (dec(m_0), dec(m_0)) \end{array}$$

In general, for $j = 0, 1$, suppose $\sqsubseteq_j$ has the form $m \sqsubseteq_j m_0$ iff $m \sqsubseteq g_j(m_0)$, where $g_j \in \mathcal{L} \to \mathcal{M}$. When $g_j(m_0) = enc(q_j)$, Equations 6 and 7 are satisfied and we say each label encodes a transition.

* * *

At this point in the text, I would like to describe a labeling that possesses $\sqsubseteq_0$ and $\sqsubseteq_1$, but does not encode transitions. I do not have an example. Do you? Could it be that there are none? How would one formalize the previous question?

4.2 Protocols

Protocols have not been mentioned, except by way of the example in Section 3.2. Here is one possible use. The labeled transition relation in Section 4.1 has the property that both $(\mathsf{bt}, (btl, enc(\mathsf{bt})), enc(\mathsf{bt}))$ and $(\mathsf{bt}, (obl, enc(\mathsf{bt})), enc(\mathsf{bt}))$ are in the transition relation. If a bundle $\mathcal{B}$ contains one event of the form $\circ(\mathsf{bt}, \mathsf{bt})$, there are two strand spaces with labels bundles faithful to it in the sense of Lemma 7. If the bundles are runs of a protocol, the role assignment might eliminate one of the two possible translations.

5 Guarded Transitions

A transition relation $t \in \mathcal{T}$ is *total* if $\forall q_0 \in \mathcal{Q}. \exists q_1 \in \mathcal{Q}. (q_0, q_1) \in t$, that is, there is a state that follows every state. The TPM transition relation in Section 3 is total.

In this section, we explore an example in which the transition relation is not total. The lack of totality causes a problem when defining the roles of its protocol. A synchronization event might not be in the transition relation.

The problem is solved by modifying a synchronization event to include a guard and amending Equation 2. With $\mathbb{B} = \{\mathfrak{t}, \mathfrak{f}\}$, a guarded state synchronization event is $\circ g$, where $g \in \mathcal{G}(\mathcal{T})$, $\mathcal{G}(\mathcal{T}) \subseteq \mathbb{B} \times \mathcal{Q} \times \mathcal{Q}$, and

$$(b, q_0, q_1) \in \mathcal{G}(\mathcal{T}) \text{ iff } b \supset (q_0, q_1) \in \mathcal{T}.$$

Equation 2 is amended to require that the guard be true in a path of a bundle.

$$\forall n \in \mathcal{N}^\circ(\mathcal{S}). evt_{\mathcal{S}}(n) = \circ(\mathfrak{t}, \pi(f(n)), \pi(f(n) + 1)). \tag{8}$$

In all other cases, the guard is ignored. A motivating example is the Wrap-Decrypt Protocol.

5.1 Wrap-Decrypt State

The state in the Wrap-Decrypt Protocol is a device that creates, stores, and shields symmetric keys. The device offers two operations using the keys it stores. It can encrypt a key using a key in the store, called wrapping, or it can decrypt a message using a key.

A goal of this device is that all of its keys remain shielded within it. A key could be leaked if a key is used to wrap itself, and then the wrapped key is decrypted. We prove this attack is impossible using strand spaces with guarded transitions. But first, we naively assume guards are unnecessary.

The device enforces its security policy by associating an attribute with each of its keys. A key has one of three attributes, init, wrap, and decrypt. A key is created with attribute init, wrapping is allowed when a key has attribute wrap, and decrypting is allowed when a key has attribute decrypt.

Attributes can be changed with the set wrap and set decrypt operation. The device policy is that set wrap succeeds as long as key's attribute is not decrypt and set decryt succeeds as long as key's attribute is not wrap. The remaining available operation of the device is key making.

The state of the device is modeled by a finite sequence of attributes. For state q, the allocated addresses of q are in $\mathcal{A}(q) = Dom(q)$, the domain of q. The key associated with address a is s_a. The handle of the key at a is $\#\,\mathsf{s}_a$. The initial state is the empty sequence.

The transition relation $\mathcal{T}$ is

$$
\begin{array}{lll}
(q_0, q_1) \in \mathcal{T} \text{ iff } q_1 = q_0 \cdot \langle \mathsf{init} \rangle & & \text{(make)} \\
\quad \vee\, \exists a \in \mathcal{A}(q_0).q_0(a) \neq \mathsf{decrypt} \wedge q_1 = q_0[a \leftarrow \mathsf{wrap}] & & \text{(set wrap)} \\
\qquad\qquad \vee\, q_0(a) \neq \mathsf{wrap} \wedge q_1 = q_0[a \leftarrow \mathsf{decrypt}] & & \text{(set decrypt)} \\
\qquad\qquad \vee\, q_0(a) = \mathsf{wrap} \wedge q_1 = q_0 & & \text{(wrap)} \\
\qquad\qquad \vee\, q_0(a) = \mathsf{decrypt} \wedge q_1 = q_0 & & \text{(decrypt),}
\end{array}
$$

where $q_0 \cdot \langle \mathsf{init} \rangle$ is the result of appending attribute init to the end of q_0 and $q_0[a \leftarrow \mathsf{wrap}]$ is the result of updating q_0 so that wrap is at address a.

The encoding function is the concatenation of all of the handles in the state. Thus, handles are carried but not their keys. In this case, the encoding function is not injective.

There is only one address for each key in a state.

$$\begin{array}{l}\forall q \in \mathcal{Q}, k : \mathsf{S}, a_0, a_1 \in \mathcal{A}(q). \\ \quad k = \mathsf{s}_{a_0} \wedge k = \mathsf{s}_{a_1} \supset a_0 = a_1.\end{array} \tag{9}$$

Let $hk(q,k)(a) = a \in \mathcal{A}(q) \wedge k = \mathsf{s}_a$. In words, $hk(q,k)$ is a predicate on addresses that is true when the address is an address of key k in state q. Using hk, Equation 9 becomes

$$\begin{array}{l}\forall q \in \mathcal{Q}, k : \mathsf{S}, a_0, a_1 \in \mathbb{N}. \\ \quad hk(q,k)(a_0) \wedge hk(q,k)(a_1) \supset a_0 = a_1.\end{array} \tag{10}$$

A useful way to exploit this property is via the epsilon operator ϵ. For unary predicate p, ϵp is a logical term that obeys the axiom $(\exists x.\, p(x)) \supset p(\epsilon p)$.

Lemma 9 (Unique Key Address)**.**

$$\begin{array}{l}\forall q \in \mathcal{Q}, k : \mathsf{S}, a \in \mathbb{N}. \\ \quad hk(q,k)(a) \supset a = \epsilon(hk(q,k)).\end{array}$$

Useful properties about paths through state space are given by the State Length Nondecreasing and the Sticky Attribute Lemmas.

Lemma 10 (State Length Nondecreasing)**.**

$$\forall \pi, i, j \in \mathbb{N}.\, i \leq j \supset |\pi(i)| \leq |\pi(j)|.$$

This lemma is used to show that when a key exists in a state in a path, it will exists in all later states.

Lemma 11 (Sticky Attribute)**.**

$$\begin{array}{l}\forall \pi, i, j \in \mathbb{N}, a \in \mathcal{A}(p(i)), r \in \mathcal{C}. \\ \quad i \leq j \wedge \pi(i)(a) = r \wedge r \neq \mathsf{init} \\ \qquad \supset \pi(j)(a) = r\end{array}$$

where $\mathcal{C}$ is the set of attributes.

$$
\begin{aligned}
\mathit{make}(q:\mathsf{Q}) &= \circ(\mathsf{t}, q, q \cdot \langle \mathsf{init} \rangle) \Rightarrow +\#\,\mathsf{s}_{|q|} \\
\mathit{setwrap}(q:\mathsf{Q}, k:\mathsf{S}) &= \begin{cases} \circ(\mathsf{t}, q, q') & \exists a.\, hk(q,k)(a) \wedge q(a) \neq \mathsf{decrypt} \\ \circ(\mathsf{f}, q, q) & \text{otherwise} \end{cases} \\
&\quad \text{where } q' = q[\epsilon(hk(q,k)) \leftarrow \mathsf{wrap}] \\
\mathit{setdecrypt}(q:\mathsf{Q}, k:\mathsf{S}) &= \begin{cases} \circ(\mathsf{t}, q, q') & \exists a.\, hk(q,k)(a) \wedge q(a) \neq \mathsf{wrap} \\ \circ(\mathsf{f}, q, q) & \text{otherwise} \end{cases} \\
&\quad \text{where } q' = q[\epsilon(hk(q,k)) \leftarrow \mathsf{decrypt}] \\
\mathit{wrap}(q:\mathsf{Q}, k_0, k_1:\mathsf{S}) &= -\#k_0 \Rightarrow -\#k_1 \Rightarrow \circ(b, q, q) \Rightarrow +\{\!|k_0|\!\}_{k_1} \\
&\quad \text{where } b = \exists a.\, hk(st, k_1)(a) \\
&\qquad\qquad \wedge\, q(a) = \mathsf{wrap} \\
\mathit{decrypt}(q:\mathsf{Q}, m:\mathsf{M}, k:\mathsf{S}) &= -\{\!|m|\!\}_k \Rightarrow -\#k \Rightarrow \circ(b, q, q) \Rightarrow +m \\
&\quad \text{where } b = \exists a.\, hk(st, k)(a) \\
&\qquad\qquad \wedge\, q(a) = \mathsf{decrypt}
\end{aligned}
$$

Figure 6: Wrap-Decrypt Traces

5.2 Wrap-Decrypt Protocol

We first explore the wrap role without guarded transitions. The *wrap* trace receives the handles of two keys and transmits the encryption of one by the other. There is a state synchronization event between the receptions and the transmission during which the state is observed.

$$\mathit{wrap}(q:\mathsf{Q}, k_0, k_1:\mathsf{S}) = -\#k_0 \Rightarrow -\#k_1 \Rightarrow \circ(q, q) \Rightarrow +\{\!|k_0|\!\}_{k_1}$$

A problem with this role is there is no way to guarantee that the synchronization event in each instance of the role is in the transition relation. As a result, there is no means to restrict instances to the case in which key k_1 is in the state q, and the attribute of key k_1 is wrap.

Figure 6 displays the Wrap-Decrypt Traces using guarded transitions. The guard in the *wrap* trace addresses both issues listed above. The *wrap* trace allows the wrapping of keys that are not in the store. We chose to simplify the trace in this way because the simplification has no impact on the security goal to be proved in Section 5.3.

The *setwrap* trace motivates the use of the ϵ operator. The context in which ϵ occurs ensures the conclusion of the ϵ axiom is implied.

In the full version of strand spaces [3], origination assumptions can be inherited from roles. In the PVS version of Wrap-Decrypt Protocol, every instantiation of the *make* role adds the assumption that the handle of the freshly created key uniquely originates at its node.

5.3 Wrap-Decrypt Security Goal

The security goal we wish to prove is that a key cannot be used both for wrapping and decryption. The goal is formalized within this framework by

Theorem 12 (Wrap-Decrypt Security Goal)**.**

$$\begin{array}{l}\forall z_0, z_1 : \mathsf{Z}, h_0, h_1 \in \mathbb{N}, q_0, q_1 : \mathsf{Q}, k_0, k_1 : \mathsf{S}, m : \mathsf{M}. \\ \quad h_0 \geq 3 \wedge h_1 \geq 3 \\ \qquad \wedge \mathsf{htin}(z_0, h_0, \mathit{wrap}(q_0, k_0, k_1)) \\ \qquad \wedge \mathsf{htin}(z_1, h_1, \mathit{decrypt}(q_1, m, k_1)) \\ \qquad\quad \supset \mathit{falsehood}\end{array}$$

In words, there is no bundle in which a wrap and a decrypt strand both include a transition node and share the key k_1. In contrast to Section 3.2, we prove the security goal is satisfied in all bundles, not just the ones that are runs of the protocol.

Proof Sketch. The general approach is to apply the Sticky Attribute Lemma (Lemma 11) to show that when the wrap node is before the decrypt node, the decrypt strand is impossible because the key's attribute is wrap, and vice versa. The steps in the proof follow. In these proofs, for event $e = \circ(b, q_0, q_1)$, let $e \downarrow 0 = b$, $e \downarrow 1 = q_0$, and $e \downarrow 2 = q_1$.

1. Prove a bridge lemma for the Sticky Attribute Lemma analogous to the Init Extend Bridge Lemma (Lemma 5).

$$\begin{array}{l}\forall n_0, n_1 \in \mathcal{N}^\circ(\mathcal{S}_\mathcal{B}), a \in \mathcal{A}(\mathit{evt}_{\mathcal{S}_\mathcal{B}}(n_0) \downarrow 2), r \in \mathcal{C}. \\ \quad n_0 \prec n_1 \wedge \mathit{evt}_{\mathcal{S}_\mathcal{B}}(n_0) \downarrow 2(a) = r \wedge r \neq \mathsf{init} \\ \qquad \supset \mathit{evt}_{\mathcal{S}_\mathcal{B}}(n_1) \downarrow 1(a) = r\end{array}$$

2. Prove a modification of the bridge lemma in which addresses are replaced by keys.

$$\begin{array}{l}\forall n_0, n_1 \in \mathcal{N}^\circ(\mathcal{S}_\mathcal{B}), k : \mathsf{S}, r \in \mathcal{C}. \\ \quad \text{let } p = hk(evt_{\mathcal{S}_\mathcal{B}}(n_0) \downarrow 2, k) \text{ in} \\ \quad n_0 \prec n_1 \wedge (\exists a.\, p(a)) \wedge evt_{\mathcal{S}_\mathcal{B}}(n_0) \downarrow 2(\epsilon p) = r \wedge r \neq \mathsf{init} \\ \qquad \supset evt_{\mathcal{S}_\mathcal{B}}(n_1) \downarrow 1(\epsilon p) = r\end{array}$$

3. Prove three lemmas about wrap strands in bundles. Prove the key k_1 is in the state, the strand is a state observer, and the attribute of k_1 is wrap.

4. Prove the analogous three lemmas about decrypt strands in bundles.

5. To the security goal, instantiate the lemmas about wrap and decrypt strands, apply the lemma that the transition nodes $(z_0, 2)$ and $(z_1, 2)$ are totally ordered, and then reason by cases.

   Case $(z_0, 2) = (z_1, 2)$**:** When $z_0 = z_1$, the attribute of k_1 in the state at $(z_0, 2)$ cannot be both wrap and $\mathsf{decrypt}$, proving *falsehood*.

   Case $(z_0, 2) \prec (z_1, 2)$**:** Instantiate the modified bridge lemma with $n_0 \mapsto (z_0, 2)$, $n_1 \mapsto (z_1, 2)$, $k \mapsto k_1$, and $r \mapsto \mathsf{wrap}$. The state at $(z_1, 2)$ cannot be both wrap and $\mathsf{decrypt}$, proving *falsehood*.

   Case $(z_1, 2) \prec (z_0, 2)$**:** Proof is analogous to the previous case. □

5.4 Guards and State Threading

Guarded transitions suggest another way to do state threading when the encoding function is injective. State threading [4] is a method for reasoning about state in a protocol without the use of state synchronization events. Instead, receive-send pairs of nodes are annotated by a subset of the transition relation. With hindsight, a better approach might have been to annotate the node pair with a guarded transition.

For the extend role in Section 3.2, the annotation in [4] for extending the state encoded by m_0 with m_1 is

$$\{(q_0, q_1) \mid q_0 = enc(m_0) \wedge q_1 = \mathsf{ex}(m_1, q_0)\}.$$

Let $p(m)(q) = (q = enc(m))$, that is, $p(m)$ is a predicate on states that is true when the state is encoded by m. With guarded transitions, the annotation becomes

$$(\exists q.\, p(m_0)(q), \epsilon(p(m_0)), \mathsf{ex}(m_1, \epsilon(p(m_0)))).$$

The advantage of using guarded transitions becomes apparent when considering bridge lemmas. The state threading version of the Init Extend Bridge Lemma is

$$\begin{aligned} n \in \mathcal{N}^\circ(\mathcal{B}) \wedge (\exists q.\, p(m_0)(q)) \wedge anno_{\mathcal{B}}(n) \downarrow 1 = \mathsf{ex}(m, \epsilon(p(m_0))) \\ \supset \exists n'.\, n' \prec n \wedge anno_{\mathcal{B}}(n') = (\mathsf{t}, \epsilon(p(m_0)), \mathsf{ex}(m, \epsilon(p(m_0)))). \end{aligned}$$

5.5 Guards and the $\mathcal{B}$, $\mathcal{C}$, ϕ Model

* * *

I don't know how to relate strand spaces with guarded transitions to some version of the $\mathcal{B}$, $\mathcal{C}$, ϕ model. If it can be done, the Wrap-Decrypt Protocol and its security goal will be easily transferred into this $\mathcal{B}$, $\mathcal{C}$, ϕ model.

6 Conclusion

The conclusion has not been written yet. Some observations follow.

- This paper sorely needs the kind of example described at the end of Section 4.1 or a proof that there are none.
- The relationship between guarded transitions and the $\mathcal{B}$, $\mathcal{C}$, ϕ model should be explored.
- Messages need not be part of an algebra to present and compare strand spaces with transition to strand spaces with labels. This is because skeletons and their homomorphisms do not explicitly occur in this discussion.
- Protocols play a surprisingly small role in this paper.
- The use of an automated proof assistant and carefully worked out examples is crucial to ensuring the models are applicable.

Acknowledgments

Thanks to Daniel J. Dougherty, Joshua D. Guttman, Moses D. Liskov, and Paul D. Rowe.

References

[1] Joseph A. Goguen and José Meseguer. Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105(2):217–273, 1992.

[2] Joshua D. Guttman. State and progress in strand spaces: Proving fair exchange. *Journal of Automated Reasoning*, 48(2):159–195, 2012.

[3] John D. Ramsdell. Proving security goals with shape analysis sentences. Technical Report MTR130488, The MITRE Corporation, September 2013. `http://arxiv.org/abs/1403.3563`.

[4] John D. Ramsdell, Daniel J. Dougherty, Joshua D. Guttman, and Paul D. Rowe. A hybrid analysis for security protocols with state. The MITRE Corporation and Worcester Polytechnic Institute, April 2014. `http://arxiv.org/abs/1404.3899`.

Contents