Choreographies with Secure Boxes and Compromised Principals

Marco Carbone*
IT University of Copenhagen
Denmark
carbonem@itu.dk

Joshua Guttman
Worcester Polytechnic Institute
United States
guttman@wpi.edu

We equip choreography-level session descriptions with a simple abstraction of a security infrastructure. Message components may be enclosed within (possibly nested) "boxes" annotated with the intended source and destination of those components. The boxes are to be implemented with cryptography.

Strand spaces provide a semantics for these choreographies, in which some roles may be played by compromised principals. A *skeleton* is a partially ordered structure containing local behaviors (strands) executed by *regular* (non-compromised) principals. A skeleton is *realized* if it contains enough regular strands so that it could actually occur, in combination with any possible activity of compromised principals. It is *delivery guaranteed (DG) realized* if, in addition, every message transmitted to a regular participant is also delivered.

We define a novel transition system on skeletons, in which the steps add regular strands. These steps solve tests, i.e. parts of the skeleton that could not occur without additional regular behavior.

We prove three main results about the transition system. First, each minimal DG realized skeleton is reachable, using the transition system, from any skeleton it embeds. Second, if no step is possible from a skeleton $\mathbb{A}$, then $\mathbb{A}$ is DG realized. Finally, if a DG realized $\mathbb{A}'$ is accessible from $\mathbb{A}$, then $\mathbb{A}'$ is minimal. Thus, the transition system provides a systematic way to construct the possible behaviors of the choreography, in the presence of compromised principals.

1 Introduction

Distributed transactions are increasingly central to our economic and social infrastructure. Rigorous, type-based notions of session are thus subjects of intense exploration, as they can ensure that communications among principals are properly coordinated [14, 11, 12, 2, 13]. However, sessions require a security infrastructure, since the data they carry may be sensitive, and a transaction may (for instance) transfer money from one person to another. Standard security infrastructures, such as TLS [7] for web interactions, are two-party, point-to-point mechanisms. When a transaction involves more than two parties—for instance, a buyer, a seller, and a bank—then it is hard to see how to use TLS sessions to ensure that the parties get any security guarantees.

An alternative—given a session choreography—is to synthesize a security infrastructure that is appropriate to the goals of that session [5, 6]. This infrastructure is effectively a custom cryptographic protocol generated specifically to ensure that malicious principals cannot undermine the behavior that the advertised session choreography promises to compliant principals. Generating this protocol, and ensuring its correctness, requires reasoning at several levels, including both the choreography level and the cryptographic level.

*The first author is partially funded by the CosmoBiz project. The second author is partially funded by the National Science Foundation, (grant no. CNS-0952287).

ICE
EPTCS , 2009, pp. 1–15.

In this paper we study reasoning specifically at the choreography level, without introducing the complexities of realistic cryptography. These complexities include selection of public-key and symmetric cryptographic primitives, as well as key distribution. Another recent paper which treats protocols by an abstraction of their cryptographic mechanisms is [1].

We use a simple choreography-level specification for security of parts of messages, which we call *boxes*. A box $[M]_{\rho_1\rho_2}$ represents the fact that message M will be sent in some format x such that, if ρ_1 and ρ_2 are uncompromised roles, then x was prepared only by ρ_1 and can be opened only by ρ_2. Boxes may appear nested inside other boxes. Naturally, any implementation of boxes will require cryptography. We might implement boxes by message structures in which ρ_1, ρ_2 first agree on a shared secret, and then use it to encrypt and provide message authentication for M (and other messages as determined by the choreography). The first step of agreeing on a shared secret may rely on public-key cryptography. Boxes are a mechanism to specify when a message component achieves secrecy and integrity between two uncompromised principals, despite other compromised principals behaving unpredictably or maliciously.

In this paper, we will develop a method to define the possible behaviors of a choreography as a function of a choice of compromised roles R. That is, given an assumption that principals not in R will behave in accordance with their roles in the choreography, we would like to define all possible behaviors a choreography execution can exhibit. To do so, we translate each choreography description into a set of strands. Each of these strands represents a possible local behavior of one principal in a single session, running a role of the choreography. These *regular*, non-compromised strands may interact with each other and with any behavior within the power of the adversary, to produce a variety of global executions. We give a method for generating all of these global executions, or more precisely, for finding the minimal, essentially different executions.

We call these minimal, essentially different executions *shapes*. Each shape is a shape *relative* to some starting point, typically some assumed local strand representing a behavior of a single participant. The shapes describe the possible *explanations* for the experience of this participant, i.e. what other local executions (strands) of regular participants would be needed in possible runs, in combination with adversary actions. They are minimal in that no lesser amount of regular behavior would yield a full explanation of regular activity in the starting point.

We generate shapes via a transition system defined by two rules. One rule says that additional strands must be added when a participant receives a box that the adversary could not create, and which is not yet explained by an earlier transmission from an uncompromised strand. It also applies to situations where a box has been removed from nested boxes, and only regular strands can extract it.

The other rule corresponds to the usual choreography assumption on the communication medium. This assumption is that the medium is *resilient*, i.e. that when an uncompromised participant sends a message to another uncompromised participant, then that message will be delivered. Since we work in a partially ordered execution model, there is no assumption about *when* this message will be delivered, relative to causally unrelated actions. We present three main results.

1. In the transition system defined by our two rules, and relative to a chosen assumption R about compromised roles, if $\mathbb{A}'$ is any shape compatible with a starting point $\mathbb{A}$, then $\mathbb{A} \longrightarrow^* \mathbb{A}'$. The same holds for shapes with guaranteed delivery. (Thm. 1.)
2. When we start from a single strand $\mathbb{A}$, then any maximal trace $\mathbb{A} \longrightarrow_S^* \mathbb{A}' \not\rightarrow$ terminates with a shape $\mathbb{A}'$ with delivery guaranteed. (Thm. 2.)
3. Every trace starting from a single strand terminates. (Thm. 3.)

In particular, the first point holds for all strand spaces based on boxes, while the second and third are specific to strand spaces defined as the semantics of choreographies in a particular syntax.

2 Abstract Strand Spaces

2.1 Basic Definitions

Definition 1 (Messages and Boxes)**.** *Messages M and boxes b are defined:*

$$W ::= \quad v \mid b \qquad\qquad M ::= \quad \tilde{W} \qquad\qquad b ::= \quad [\tilde{M}]_{\rho_1\rho_2}$$

where $\tilde{\cdot}$ denotes a tuple of zero or more elements. v is a basic value—belonging to a finite *set of basic values—and ρ_i ranges over the set of roles $\mathscr{R}$.*

A message M can either be a value v or a box $[\tilde{M}]_{\rho_1\rho_2}$. We also use letter c to denote boxes. A box is a tuple of messages M_i from ρ_1 that can only be opened by ρ_2.

A *strand space*, first introduced in [15] as a formalism for reasoning about cryptographic protocols, is a collection of strands. Here, we introduce *abstract strand spaces*, strand spaces where messages range over M (unlike the original version with cryptography). A *substitution* is a function that maps basic values to basic values. Since basic values form a finite set, there are only finitely many substitutions.

Definition 2 (Abstract Strand Space)**.** *A* directed term *is a pair denoted by $\pm M$ where $\pm \in \{-,+\}$ is a direction with $+$ representing transmission and $-$ reception. A* trace *is an element of $(\pm M)^*$, the set of finite sequences of directed terms.*
An abstract strand space *is a set S with a trace mapping $\mathsf{tr} : S \to (\pm M)^*$. A* strand *is an element of S.*
A strand space S is closed under *a set of substitutions Σ, if, for every $s \in S$ and $\sigma \in \Sigma$, there is an $s' \in S$ such that $\mathsf{tr}(s') = \sigma(\mathsf{tr}(s'))$.*

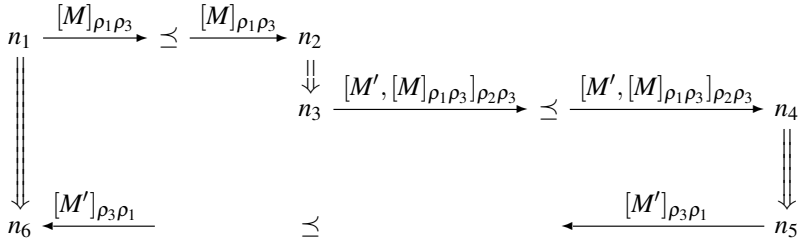
In this paper we consider *finite* strand spaces that are closed under substitutions of basic values for basic values.

Notation. If $s \in S$ is a strand then $s(i)$ denotes the i^{th} element of the trace of s and is called *node*. We write $m \Rightarrow n$ when n is the node immediately after m on the same strand s i.e. $m = s(i)$ and $n = s(i+1)$. Also, $\mathsf{msg}(n)$ denotes the message of the directed term in n while $\mathsf{neg}(n)$ ($\mathsf{pos}(n)$) holds if n is a reception (transmission) node.
It is now interesting to see how these input/output traces could be combined together in order to form a real execution. *Skeletons* express parts of an execution (with some pending transmission/reception nodes related to adversary activity):

Definition 3 (Skeleton)**.** *Given a strand space S, a* skeleton $\mathbb{A}$ *is a finite set of regular nodes (nodes belonging to strands of S), denoted by* $\mathsf{nodes}(\mathbb{A})$*, equipped with a partial order $\preceq_{\mathbb{A}}$ on* $\mathsf{nodes}(\mathbb{A})$ *indicating causal precedence (consistent with $\Rightarrow$). Moreover, if $m \Rightarrow n$ and $n \in \mathsf{nodes}(\mathbb{A})$, then $m \in \mathsf{nodes}(\mathbb{A})$.*

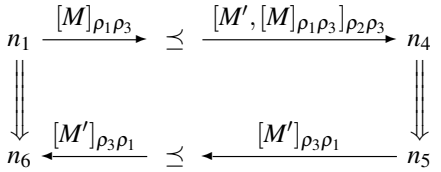
In the rest of the paper, $\prec$ will denote the non-reflexive subrelation of $\preceq$.

Example 1. As an example, let us consider a skeleton composed by three strands. Below, outgoing and incoming edges denote transmission and reception nodes respectively.



(1)

The three strands above belong to roles ρ_1, ρ_2 and ρ_3 respectively. If the middle strand was not there e.g. if ρ_2 were compromised, then we would have the following skeleton:



As previously said, some roles may belong to compromised principals. In the sequel, we set $R \subseteq \mathscr{R}$ to be the set of compromised roles. Moreover, we assume that each strand is always marked with the role it belongs to (a strand belongs to exactly one role). On this premises, it is natural to define the untamed behaviour of R (or adversary) in terms of *penetrator* strands:

Definition 4 (Abstract Penetrator)**.** *$\mathscr{P}_R$, the* abstract penetrator *for a set of compromised roles R, is the set of strands of the forms:*

$$\textbf{(C)} \quad -M_0 \Rightarrow -(M_1,\ldots,M_k) \Rightarrow +(M_0,M_1,\ldots,M_k) \qquad \textbf{(A)} \quad +\tilde{v}$$
$$\textbf{(S)} \quad -(M_0,M_1,\ldots,M_k) \Rightarrow +M_0 \Rightarrow +(M_1,\ldots,M_k)$$
$$\textbf{(B)} \quad -\tilde{M} \Rightarrow +[\tilde{M}]_{\rho_1\rho_2} \quad \textit{where } \rho_1 \in R$$
$$\textbf{(O)} \quad -[\tilde{M}]_{\rho_1\rho_2} \Rightarrow +\tilde{M} \quad \textit{where } \rho_2 \in R$$

Above, (**C**) allows the penetrator to *compose* received messages and resend them; in (**S**), a compound message can be *separated* and resent; (**B**) allows to *box* messages and sign them with a compromised role (from R); with (**O**), the penetrator can *open* boxes targeted to compromised roles; and using (**A**), the penetrator can send any clear text.

We can compose (instances of) the various strands above with a skeleton in order to build the *graph of interaction* of a skeleton $\mathbb{A}$ with respect to a penetrator $\mathscr{P}_R$ i.e. an acyclic directed graph $\mathscr{B}$ whose nodes are the nodes of strands in $\mathscr{P}_R$ and $\mathbb{A}$, and whose edges can be obtained by connecting any transmitting node m with a receiving node n such that $\mathsf{msg}(m) = \mathsf{msg}(n)$. We say of two nodes m_0, m_k of $\mathscr{B}$ that $m_0 \preceq_{\mathscr{B}} m_k$ if there is a sequence $m_0, m_1, \ldots, m_k$ such that for each pair m_i, m_{i+1}, either $m_i \preceq_{\mathbb{A}} m_{i+1}$, or $m_i \Rightarrow^+ m_{i+1}$ on a penetrator strand of $\mathscr{P}_R$, or m_i is a transmission node and m_{i+1} is a receiving node connected to it.

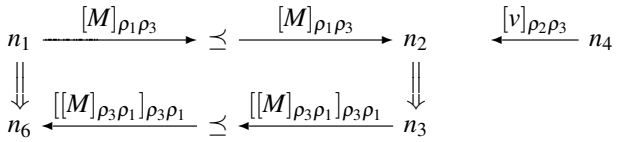
We now define a *realized skeleton* i.e. a skeleton that has precisely the behavior of some execution:

Definition 5 (Realized Skeleton)**.** *A skeleton $\mathbb{A}$ is* realized *if there is a graph of interaction $\mathscr{B}$ of $\mathbb{A}$ wrt $\mathscr{P}_R$ such that every reception node has an incoming edge, and for all nodes $m, n \in \mathsf{nodes}(\mathbb{A})$, $m \preceq_{\mathscr{B}} n$ implies $m \preceq_{\mathbb{A}} n$.*

A *shape* is a minimal homomorphism preserving $\preceq$ that maps a skeleton into a realized one. Below, a homomorphism $H : \mathbb{A}_0 \mapsto \mathbb{A}_1$ is node-wise injective if it is an injective function on the nodes of $\mathbb{A}_0$. Moreover, H_0 is node-wise less than or equal to H_1, written $H_0 \leq H_1$, if for some node-wise injective L, $L \circ H_0 = H_1$. We then say that H_0 is node-wise minimal in some set Z whenever $H_0 \in Z$ and for any $H \in Z$, $H \leq H_0$ implies H and H_0 are isomorphic.

Definition 6 (Shape [8])**.** *$H : \mathbb{A}_0 \mapsto \mathbb{A}'$ is a shape for $\mathbb{A}_0$ if H is node-wise minimal among the set of homomorphisms $H' : \mathbb{A}_0 \mapsto \mathbb{A}''$ where $\mathbb{A}''$ is realized.*

Sometimes, with an abuse of terminology, if $H : \mathbb{A} \mapsto \mathbb{A}'$ is a shape for $\mathbb{A}$, we shall say that $\mathbb{A}'$ is a shape for $\mathbb{A}$. For instance, the skeleton in (1) is a shape for $n_1 \Rightarrow n_6$. On the other hand, because of the extra node n_4, the following realized skeleton is not a shape for $n_1 \Rightarrow n_6$:



(2)

We also consider special skeletons which guarantee that messages are delivered.

Definition 7 (Delivery-Guaranteed Skeletons)**.** *A* delivery-guaranteed skeleton *(DG skeleton) is a skeleton such that for every positive node n such* $\mathsf{msg}(n) = [\tilde{M}]_{\rho_1\rho_2}$ *and* $\rho_2 \notin R$ *there exists a negative node* n' *on another strand such that* $\mathsf{msg}(n) = \mathsf{msg}(n)'$.

Note that (2) is not DG while (1) is. Delivery-guaranteed skeletons characterize some special shapes:

Definition 8 (Delivery-Guaranteed Shape)**.** $H : \mathbb{A}_0 \mapsto \mathbb{A}'$ *is a DG shape for* $\mathbb{A}_0$ *if H is node-wise minimal among the set of homomorphisms* $H' : \mathbb{A}_0 \mapsto \mathbb{A}''$ *where* $\mathbb{A}''$ *is a realized and DG skeleton.*

2.2 Characterizing Realized Skeletons

In this subsection, we will introduce a characterization of realized skeletons in the spirit of [9]. The idea is to use *authentication tests* [8] as a method for explaining why a message is suddenly found outside a box which was previously containing it. In general, either the box owner is compromised or else it was transmitted by a regular strand. The following definition formalizes the idea of a message occurring inside or outside a set of boxes.

Definition 9. *A message* M_0 *is* found only within a set of boxes B in M_1*, written* $M_0 \odot^B M_1$*, whenever every occurrence of* M_0 *in* M_1 *is nested inside a box of B.*
A message M_0 *is* found outside B in M_1*, written* $M_0 \dagger^B M_1$*, whenever not* $M_0 \odot^B M_1$.

As an example, for $M \neq$ "Hi", M is found only within $\{[M]_{\rho_1\rho_2}, [[M,\text{"}Hi\text{"}]_{\rho_3\rho_1},\text{"}Hi\text{"}]_{\rho_3\rho_4}\}$ in $[[M]_{\rho_1\rho_2}]_{\rho_2\rho_3}$ and $[[[(M,\text{"}Hi\text{"})]_{\rho_3\rho_1},\text{"}Hi\text{"}]_{\rho_3\rho_4}]_{\rho_4\rho_1}$. Also, M is found only within $\{[\text{"}Hi\text{"}]_{\rho_3\rho_1}\}$ in $[[\text{"}Hi\text{"}]_{\rho_1\rho 2}]_{\rho_3\rho_1}$ as it does not occur at all. On the contrary, M is found outside $\{[\text{"}Hi\text{"}]_{\rho_1\rho_2}\}$ in $[M,[\text{"}Hi\text{"}]_{\rho_1\rho_2}]_{\rho_3\rho_4}$.

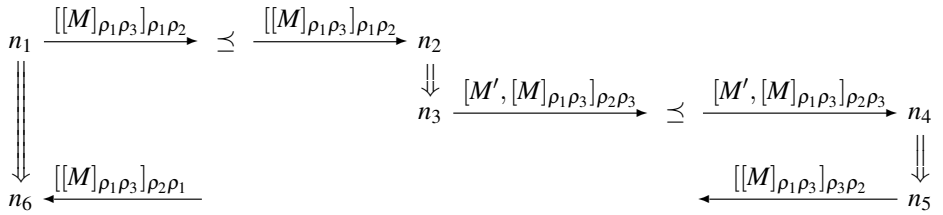
Given a skeleton, a set of boxes and a message, we can highlight those minimal nodes for which such a message is found only outside the boxes. This is formalized by the notion of cut:

Definition 10 (Cut)**.** *Let M be a message, B a set of boxes and* $\mathbb{A}$ *a skeleton. Then,*

$$\mathit{Cut}(M,B,\mathbb{A}) = \{n \in \mathit{nodes}(\mathbb{A}) : \exists m \preceq_{\mathbb{A}} n \text{ and } M \dagger^B \mathit{msg}(m)\}$$

$\mathit{Cut}(M,B,\mathbb{A})$ *is defined whenever there exists a node n in* $\mathbb{A}$ *such that* $M \dagger^B \mathit{msg}(n)$.

Note that M occurs outside B in all minimal nodes of $\mathsf{Cut}(M,B,\mathbb{A})$. In the following skeleton $\mathbb{A}$,



(3)

$\mathsf{Cut}([M]_{\rho_1\rho_3}, B, \mathbb{A})$ is the set $\{n_3, n_4, n_5, n_6\}$ with minimal nodes $\{n_3, n_6\}$ for $B = \{[[M]_{\rho_1\rho_3}]_{\rho_1\rho_2}\}$ and the whole skeleton $\mathbb{A}$ for $B = \emptyset$ (assuming that $\{n_1\} \in \rho_1$, $\{n_2, n_3, n_6\} \in \rho_2$ and $\{n_4, n_5\} \in \rho_3$ and no role is in R). Also, $\mathsf{Cut}([M]_{\rho_1\rho_3}, B, \mathbb{A}) = \{n_6\}$ for $B = \{[[M]_{\rho_1\rho_3}]_{\rho_3\rho_2}\}$ but empty if ρ_2 were compromised. In the subskeleton $\mathbb{A}'$ composed by nodes n_1 and n_2 we have $\mathsf{Cut}([M]_{\rho_1\rho_3}, \{[[M]_{\rho_1\rho_3}]_{\rho_1\rho_2}\}, \mathbb{A}') = \emptyset$.

The idea behind authentication tests is that any minimal node in a cut needs to be explained in the skeleton. In other words, there must be an earlier sequence of events that extracted the message out of some box or *legally* created it. Formally,

Definition 11 (Solved Cut)**.** *A cut $\mathsf{Cut}(M, B, \mathbb{A})$ is solved wrt a set of compromised roles R, if for any of its $\prec_{\mathbb{A}}$-minimal nodes m_1:*

1. *either m_1 is a transmission node;*
2. *or $M = [\tilde{M}]_{\rho_1\rho_2}$ and $\rho_1 \in R$, or for some $[\tilde{M}]_{\rho_1\rho_2} \in B$, $\rho_2 \in R$.*

The definition above says that a cut $\mathsf{Cut}(M, B, \mathbb{A})$ is solved whenever, for every minimal reception node n, M is outside B in n because of some penetrator activity. For instance, in (3), $\mathsf{Cut}([M]_{\rho_1\rho_3}, B, \mathbb{A}) = \{n_6\}$ is not solved for $B = \{[[M]_{\rho_1\rho_3}]_{\rho_3\rho_2}, [[M]_{\rho_1\rho_3}]_{\rho_1\rho_2}, [M, [M]_{\rho_1\rho_3}]_{\rho_2\rho_3}\}$ and $R = \emptyset$ while it is solved if $R = \{\rho_2\}$. The above definition turns to be a crucial property of realized skeletons. In fact, the following proposition states that the property of being realized is characterized by all if its cuts being solved.

Proposition 1. *Let $\mathbb{A}$ be a skeleton. Then, every cut in $\mathbb{A}$ is solved if and only if $\mathbb{A}$ is realized.*

Proof. $\Rightarrow$:

We prove this by contradiction. Assume that $\mathbb{A}$ is not realized. Then, by definition, there must be an input node n containing a message that a penetrator $\mathscr{P}_R$ is not allowed to send i.e. there is some node n such that, for all $m \prec_{\mathbb{A}} n$, either (i) $\rho_1 \notin R$, $[\tilde{M}]_{\rho_1\rho_2}$ is nested in $\mathsf{msg}(n)$ and does not occur in $\mathsf{msg}(m)$; or (ii) for some message M and $\rho_2 \notin R$, we have that $M \dagger^{\{[\tilde{M}]_{\rho_1\rho_2}\}} \mathsf{msg}(n)$ and $M \odot^{\{[\tilde{M}]_{\rho_1\rho_2}\}} \mathsf{msg}(m)$. If (i) holds, then $\mathsf{Cut}([\tilde{M}]_{\rho_1\rho_2}, \emptyset, \mathbb{A})$ is clearly unsolved. Similarly, if (ii) then $\mathsf{Cut}(M, \{[\tilde{M}]_{\rho_1\rho_2}\}, \mathbb{A})$ is unsolved.

$\Leftarrow$: Assume that there is a cut $\mathsf{Cut}(M, B, \mathbb{A})$ which is not solved. That means, that there is a minimal input node where M is only found inside B and such that $M \neq [\tilde{M}]_{\rho_1\rho_2}$ for $\rho_1 \in R$, and for no $[\tilde{M}]_{\rho_1\rho_2} \in B$, $\rho_2 \in R$. But then, there is no penetrator activity which could derive M hence $\mathbb{A}$ would not be realized. □

We conclude this section observing that an unsolved cut implies the existence of another unsolved cut whose boxes B are messages appearing in the current skeleton. In the sequel, let the relation $M \sqsubset M'$ hold whenever M is contained in M' ($\sqsubseteq$ is the reflexive closure).

Proposition 2. *Let $\mathbb{A}$ be a skeleton and $\mathsf{Cut}([M]_{\rho_1\rho_2}, B, \mathbb{A})$ an unsolved cut. Then, $\mathsf{Cut}([M]_{\rho_1\rho_2}, B', \mathbb{A})$ is also unsolved for $B' = \{b \mid n' \prec_{\mathbb{A}} n \text{ s.t. } [\tilde{M}]_{\rho_1\rho_2} \sqsubset b \sqsubseteq \mathsf{msg}(n') \wedge \mathsf{rcv}(b) \notin R\}$ for some $\preceq_{\mathbb{A}}$-minimal input node n in $\mathsf{Cut}([M]_{\rho_1,\rho_2}, B, \mathbb{A})$ and $\rho_1 \notin R$.*

Proof. Let $c = [\tilde{M}]_{\rho_1\rho_2}$. From Definition 11, there exists a $\preceq_{\mathbb{A}}$-minimal *input* node n in $\mathsf{Cut}(c, B, \mathbb{A})$, such that $\rho_1 \notin R$ and $\rho_4 \notin R$ for all $[\tilde{M}]_{\rho_3\rho_4} \in B$. Let us now consider the predecessors n' of n in $\mathbb{A}$ which, by definition of cut, are such that $c \odot^{B} \mathsf{msg}(n')$. We consider two cases: (i) if none of n's predecessors contains c then $B' = \emptyset$ and therefore $\mathsf{Cut}(c, \emptyset, \mathbb{A})$ is unsolved as n is a minimal node such that $c \dagger^{\emptyset} \mathsf{msg}(n')$; (ii) $n_1 \ldots n_k$ are n's predecessors such that $c \sqsubset \mathsf{msg}(n_i)$. Letting

$$B_i = \{b \mid c \sqsubset b \sqsubseteq \mathsf{msg}(n_i) \text{ and } \mathsf{rec}(b) \notin R\},$$

$B_i \subseteq B$, because n is a minimal node in $\mathsf{Cut}(c, B, \mathbb{A})$. Thus, as $c \odot^{\bigcup_i B_i} \mathsf{msg}(n_i)$, n is also minimal in $\mathsf{Cut}(c, \bigcup_i B_i, \mathbb{A})$. Finally, as $B' = \bigcup_i B_i$, we can conclude that $\mathsf{Cut}(c, B', \mathbb{A})$ is also unsolved. □

3 Searching for Shapes

The results on cuts suggest a possible way of adding nodes to a skeleton so that it can become realized. We shall now address this problem and introduce a constructive method for deriving realized skeletons from non-realized ones. In the sequel, the operation $\mathbb{A} \cup \uparrow_m \textit{ with } m \prec n$, returns the skeleton $\mathbb{A}'$ consisting of $\mathbb{A}$ and the nodes $\{m' \mid m' \Rightarrow^* m \wedge m' \in \mathsf{nodes}(S)\}$, with the ordering strengthened so that $m \prec_{\mathbb{A}'} n$. Similarly, $\mathbb{A} \cup \uparrow_m$ with $n \prec m$ is the corresponding $\mathbb{A}'$ with the opposite order enrichment $n \prec_{\mathbb{A}'} m$.

Definition 12 (Reduction Rules)**.** *The relation between skeletons $\mathbb{A} \longrightarrow_S \mathbb{A}'$, is the minimum relation satisfying the following rules:*

$$\text{(A1)}\quad \frac{\begin{array}{ccc} n \in \mathbb{A} \wedge \mathsf{neg}(n) & c = [\tilde{M}]_{\rho_1\rho_2} & c \dagger^B \mathit{msg}(n) \\ m \in S \backslash \mathbb{A} \wedge \mathit{pos}(m) & \rho_1 \notin R & c \dagger^B \mathit{msg}(m) \\ \multicolumn{3}{c}{\forall m'.\, m' \prec_{\mathbb{A}} n \vee m' \Rightarrow^+ m \quad \textit{implies} \quad c \odot^B \mathit{msg}(m')} \end{array}}{\mathbb{A} \longrightarrow_S \mathbb{A} \cup \uparrow_m \textit{ with } m \prec n}$$

$$\text{(A2)}\quad \frac{\begin{array}{lll} n \in \mathbb{A} \wedge \mathsf{pos}(n) & \mathsf{msg}(n) = [\tilde{M}]_{\rho_1\rho_2} & \neg(\exists m' \in \mathit{neg}(\mathbb{A}).\, n \prec m' \wedge \mathit{msg}(m') = \mathit{msg}(n)) \\ & \rho_2 \notin R & m \in (S) \wedge \mathsf{neg}(m) \wedge \mathit{msg}(m) = \mathit{msg}(n) \end{array}}{\mathbb{A} \longrightarrow_S \mathbb{A} \cup \uparrow_m \textit{ with } n \prec m}$$

where the set of strands S (strand space domain) is the set of regular strands. Observe in rule (A1) that if there is any B that satisfies the premise, then

$$B = \{b \mid n' \prec_{\mathbb{A}} n \;\; s.t. \;\; [\tilde{M}]_{\rho_1\rho_2} \sqsubset b \sqsubseteq \mathit{msg}(n') \wedge \mathit{rcv}(b) \notin R\}$$

We briefly comment the rules above. The first rule adds, when possible, nodes that explain why a message is found outside a box. Given a box c, the set of boxes B and a node n which is minimal in $\mathsf{Cut}(c, B, \mathbb{A})$, we choose m to be the minimal node preceding n such that c is found outside B. Note that m may already be in the skeleton (added together with some m' such that $m \Rightarrow^* m'$) and the rule still be applicable because $\preceq$ needs to be updated. The second rule deals with adding a recipient, if any is found, to an output node.

Proposition 3. *If $\mathbb{A} \longrightarrow_S \mathbb{A}'$ then $\mathbb{A}$ is not realized or $\mathbb{A}$ is not DG.*

Proof. If the reduction $\mathbb{A} \longrightarrow_S \mathbb{A}'$ is obtained by applying rule (A1), then the cut $\mathsf{Cut}([\tilde{M}]_{\rho_1\rho_2}, B, \mathbb{A})$ is clearly not solved. On the other hand, if $\mathbb{A} \longrightarrow_S \mathbb{A}'$ by (A2), then we are clearly adding an input node to a pending output. □

In the sequel, we say that a homomorphism $H : \mathbb{A} \mapsto \mathbb{A}'$ is an *augmentation* whenever H is an inclusion (identity on the domain $\mathbb{A}$), any node in $\mathbb{A}' \backslash \mathbb{A}$ belongs to the same strand and $\preceq_{\mathbb{A}'}$ is an extension of $\preceq_{\mathbb{A}}$. Directly from the rules, it follows that:

Proposition 4. *Let H map $\mathbb{A}$ to $\mathbb{A}'$ such that $\mathbb{A} \longrightarrow_S \mathbb{A}'$. Then H is an augmentation.*

Building on the above proposition, we say that H is of *type 1* (*type 2*) if it corresponds to the application of a rule 1 (rule 2).

In the sequel $\mathbb{A} \longrightarrow_S^* \mathbb{A}'$ holds whenever there exists a finite sequence $\mathbb{A}_1 \longrightarrow_S \ldots \longrightarrow_S \mathbb{A}_k$ such that $\mathbb{A} = \mathbb{A}_1$ and $\mathbb{A}' = \mathbb{A}_k$. Moreover, $\mathbb{A} \not\rightarrow$ whenever there is no $\mathbb{A}'$ such that $\mathbb{A} \longrightarrow_S \mathbb{A}'$. The following result states that we can always reach all the shapes by repeatedly applying the rules.

Theorem 1 (Completeness)**.**

1. *Let $\mathbb{A}$ be a single-strand skeleton and H a shape such that $\mathbb{A} \mapsto \mathbb{A}'$. Then $\mathbb{A} \longrightarrow_S^* \mathbb{A}'$ (up-to isomorphism).*
2. *Let $\mathbb{A}$ be a single-strand skeleton and H a DG shape such that $\mathbb{A} \mapsto \mathbb{A}'$. Then $\mathbb{A} \longrightarrow_S^* \mathbb{A}'$ (up-to isomorphism).*

Proof. From Proposition 4, we only have to prove that shapes can be expressed as the composition of augmentations of type 1 or 2 (type 2 is only considered when proving point 2). Formally, we show that there exist a k such that for every $i \in \{0,1\ldots,k\}$ we have $H = L_i \circ H_i \circ \ldots \circ H_1 \circ H_0$ where L_i is a node-wise injective homomorphism, H_0 the identity mapping and $H_1,\ldots,H_i$ augmentations.

The first step is to show how we can find k and inductively construct each H_i and L_i starting from the identity:

- **Base Case.** As H_0 must be the identity, we chose $L_0 = H$ noting that H is node-wise injective by definition of shape. We then have that $H = L_0 \circ H_0$.
- **Inductive Case.** Let $H = L_i \circ H_j \circ \ldots \circ H_1 \circ H_0$ such that H_0 is the identity and $H_1,\ldots,H_i$ are augmentations. If L_i is an isomorphism then $i = k$ and we can stop. In fact, by definition of shape, H is the minimum realized skeleton hence the image of $H_j \circ \ldots \circ H_1 \circ H_0$ is isomorphic to $\mathbb{A}'$, image of H.

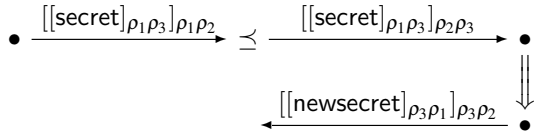
  Let L_i be not an isomorphism. Moreover, let $L_i : \mathbb{A}_j \mapsto \mathbb{A}'$ and $H_j \circ \ldots \circ H_1 \circ H_0 : \mathbb{A} \mapsto \mathbb{A}_j$ for some $\mathbb{A}_j$. We show how to construct H_{i+1} and L_{i+1}. By definition of shape, as L_i is not an isomorphism, $\mathbb{A}_j$ is not realized. If that is the case, then either there is a dangling output (this is to be considered only when proving point 2) or, by Proposition 1, there exists an unsolved cut $\mathsf{Cut}([\tilde{M}]_{\rho_1\rho_2}, B, \mathbb{A}_j)$ i.e., by definition of cut, there exists an input node m_1, $\preceq_{\mathbb{A}_j}$-minimal in $\mathsf{Cut}([M]_{\rho 1 \rho 2}, B, \mathbb{A}')$, such that $\rho_1 \notin R$ and for all $[\tilde{M}]_{\rho_3\rho_4} \in B$, $\rho_4 \notin R$. Now, as $\mathbb{A}'$ is realized, all cuts must be solved. Then, because L_i is node-wise injective, we can choose a node in the pre-image of L_i which is not in $\mathbb{A}_i$ but solves $\mathsf{Cut}([\tilde{M}]_{\rho_1\rho_2}, B, \mathbb{A}_j)$ (or add the corresponding input when proving point 2). Adding this node, precisely corresponds to an augmentation induced by rule (A1) (or (A2)) which will be our H_{i+1}. We can then choose L_{i+1} to be equal to L_i but also mapping the new added node to $\mathbb{A}'$ accordingly.

The above procedure shows how to construct the various H_i and L_i. In order to complete the proof, we need to show that we always find the k. But this follows by the fact that augmentations always increase the size of a skeleton and observing that we stop once we reach an isomorphism. □

Example 2. Let $S = \{s_i\}_{i=1,\ldots,5}$, $s_1, s_2 \in \rho_1$, $s_3, s_4 \in \rho_2$, $s_5 \in \rho_3$ and such that:

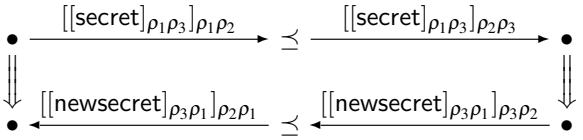
$$
\begin{aligned}
s_1 &= +[[\mathsf{secret}]_{\rho_1\rho_3}]_{\rho_1\rho_2} \Rightarrow -[\mathsf{reject}]_{\rho_2\rho_1} \\
s_2 &= +[[\mathsf{secret}]_{\rho_1\rho_3}]_{\rho_1\rho_2} \Rightarrow -[[\mathsf{newsecret}]_{\rho_3\rho_1}]_{\rho_2\rho_1} \\
s_3 &= -[[\mathsf{secret}]_{\rho_1\rho_3}]_{\rho_1\rho_2} \Rightarrow +[\mathsf{reject}]_{\rho_2\rho_1} \\
s_4 &= -[[\mathsf{secret}]_{\rho_1\rho_3}]_{\rho_1\rho_2} \Rightarrow +[[\mathsf{secret}]_{\rho_1\rho_3}]_{\rho_2\rho_3} \Rightarrow -[[\mathsf{newsecret}]_{\rho_3\rho_1}]_{\rho_3\rho_2} \Rightarrow +[[\mathsf{newsecret}]_{\rho_3\rho_1}]_{\rho_2\rho_1} \\
s_5 &= -[[\mathsf{secret}]_{\rho_1\rho_3}]_{\rho_2\rho_3} \Rightarrow +[[\mathsf{newsecret}]_{\rho_3\rho_1}]_{\rho_3\rho_2}
\end{aligned}
$$

If, for instance, $\rho_2 \in R$ and we start from s_5, we can then apply (A1) for $B = \emptyset$, $M = [\mathsf{secret}]_{\rho_1\rho_3}$ and m being the first node of the strands s_1/s_2. We obtain the following skeleton:



(4)

which is a shape for s_5. If we start from s_2, we can then apply (A1) for $B = \emptyset$, $M = [\mathsf{secret}]_{\rho_1\rho_3}$ and m being the second node of s_5. We then have:

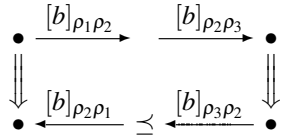


Above we have actually applied (A1) twice, where the second application just added the top $\preceq$. Note that (4) differs from the above because the latter has more information about s_2 but they are both realized (and DG).

The set of boxes B is not always empty. For instance, for $b = [\mathsf{secret}]_{\rho_1\rho_3}$, with strands

$$\begin{aligned} s'_2 &= +[b]_{\rho_1\rho_2} \Rightarrow -[b]_{\rho_2\rho_1} \\ s'_4 &= -[b]_{\rho_1\rho_2} \Rightarrow +[b]_{\rho_2\rho_3} \Rightarrow -[b]_{\rho_3\rho_2} \Rightarrow +[b]_{\rho_2\rho_1} \\ s'_5 &= -[b]_{\rho_2\rho_3} \Rightarrow +[b]_{\rho_3\rho_2} \end{aligned}$$

and applying (A1) to s'_2 with $R = \emptyset$, we get the following skeleton for $M = b$ and $B = \{[b]_{\rho_1\rho_2}\}$:



4 A Protocol Description Calculus

We illustrate our ideas with the simplest possible calculus. The syntax of this minimal choreography language (based on the Global Calculus [4]) is given by the following grammar:

$$\begin{array}{lll} C ::= & \Sigma_i \rho_1 \to \rho_2 : \mathsf{op_i}\langle \tilde{M}_i \rangle.\, C_i & \text{(interaction)} \\ \mid & \mathbf{0} & \text{(inactive)} \end{array}$$

Above, the term $\Sigma_i \rho_1 \to \rho_2 : \mathsf{op_i}\langle \tilde{M}_i \rangle.\, C_i$ describes an interaction where a branch with label op_i is non-deterministically selected and a message $\tilde{M}_i$ is sent from role ρ_1 to role ρ_2. Each two roles in a choreography share a private channel hence it would be redundant to have them explicit in the syntax [2]. Term $\mathbf{0}$ denotes the inactive system. Given a choreography C, we assume that the various op, also on different interactions, are distinct: given the lack of an iteration operator e.g. recursion, this is a constraint that can be imposed statically and we include in the well-formedness condition at the end of this section.

Our mini-language can be equipped with a standard trace semantics with configurations $C \xrightarrow{\mu} C'$ where $\mu = (\rho_1, \rho_2, \mathsf{op}_i, \tilde{M}_i)$ contains the parameters of the interaction performed i.e. $\rho_1 \to \rho_2 : \mathsf{op_i}\langle \tilde{M}_i \rangle.\, C_i \xrightarrow{(\rho_1,\rho_2,\mathsf{op}_i,\tilde{M}_i)} C_i$. A sequence of labels $\{\mu_i\}_i$ describes the temporal order in which the various described communications take place and it is called *trace*.

Assumption 1 (Well-Formedness)**.** *A choreography C is well-formed whenever:*

- *All* op*'s are distinct;*
- *let* Γ *be a set of pairs* $\rho : \tilde{M}$*. Then,* $\Gamma \vdash C$ *such that for all* ρ*,* $\Gamma(\rho)$ *has no boxes and* $\vdash$ *is defined by the following rules:*

$$\text{(T-INTERACT)}\ \frac{\tilde{M}_i \subseteq \Gamma(\rho_1) \qquad \Gamma[\rho_2 \mapsto \Gamma(\rho_2) \cup \{\tilde{M}_i\}] \vdash C_i \qquad \rho_2 \in \mathsf{top}(C_i)}{\Gamma \vdash \Sigma_i\, \rho_1 \to \rho_2 : \mathsf{op}_i\langle \tilde{M}_i \rangle.\ C_i} \qquad \text{(T-INACT)}\ \frac{}{\Gamma \vdash \mathbf{0}}$$

$$\text{(T-BOX}_1\text{)}\ \frac{\Gamma \vdash C \qquad \tilde{M} \in \Gamma(\rho_1)}{\Gamma[\rho_1 \mapsto \Gamma(\rho_1) \cup \{[\tilde{M}]_{\rho_1\rho_2}\}] \vdash C} \qquad \text{(T-BOX}_2\text{)}\ \frac{\Gamma \vdash C \qquad [\tilde{M}]_{\rho_1\rho_2} \in \Gamma(\rho_2)}{\Gamma[\rho_2 \mapsto \Gamma(\rho_2) \cup \{\tilde{M}\}] \vdash C}$$

where $\mathsf{top}(\Sigma_i\, \rho_1 \to \rho_2 : \mathsf{op}_i\langle \tilde{M}_i \rangle.\ C_i) = \{\rho_1\}$ *and* $\mathsf{top}(\mathbf{0}) = \mathscr{R}$*.*

The rules above are a simple static check for ensuring that a box $[\tilde{M}]_{\rho_1\rho_2}$ always originate by an interaction from ρ_1 and can only be opened by ρ_2 upon reception of the box (maybe nested in other boxes). An environment Γ is a function that associates a set of messages to a role. (T-INTERACT) checks $\Gamma(\rho_1)$ contains each $\tilde{M}_i$ and allows ρ_2 to use $\tilde{M}_i$ in C_i. Moreover, the rules checks that ρ_2 is the sender in C_i. (T-BOX$_1$) says that if ρ_1 knows $\tilde{M}$ then it can also create $[\tilde{M}]_{\rho_1\rho_2}$ for any ρ'. Dually, in (T-BOX$_2$), if ρ_2 knows $[\tilde{M}]_{\rho_1\rho_2}$ then it also knows $\tilde{M}$. Rule (T-INACT) allows to type $\mathbf{0}$ with any Γ.

Example 3 (Buyer-Seller Protocol)**.** Hereby, we report a Buyer-Seller financial protocol [4, 3]. A buyer Buyer asks a seller Seller for a quote about a product. If the quote is accepted, Buyer will send its credit card `card` together with the accepted quote to Seller who will forward it to a bank Bank. The bank will check if payment can be done and, if so, reply with a receipt `receipt` which will be forwarded to Buyer by Seller. In our mini-language we use boxes to make sure that the credit card number can only be read by Bank and that Seller does not change the accepted quote:

1. $\mathsf{Buyer} \to \mathsf{Seller} : \mathsf{Req}\langle \mathtt{prod} \rangle.\ \mathsf{Seller} \to \mathsf{Buyer} : \mathsf{Reply}\langle \mathtt{quote} \rangle.$
2. $(\ \mathsf{Buyer} \to \mathsf{Seller} : \mathsf{Accept}\langle [(\mathtt{quote}, \mathtt{card})]_{\mathsf{BuyerBank}} \rangle.\ \mathsf{Seller} \to \mathsf{Bank} : \mathsf{Pay}\langle (\mathtt{quote}, [(\mathtt{quote}, \mathtt{card})]_{\mathsf{BuyerBank}}) \rangle.$
3. $\quad (\ \mathsf{Bank} \to \mathsf{Seller} : \mathsf{Ok}\langle [\mathtt{receipt}]_{\mathsf{BankBuyer}} \rangle.\ \mathsf{Seller} \to \mathsf{Buyer} : \mathsf{Succ}\langle [\mathtt{receipt}]_{\mathsf{BankBuyer}} \rangle$
4. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad +$
5. $\quad\ \ \mathsf{Bank} \to \mathsf{Seller} : \mathsf{NotOk}\langle \mathsf{reason} \rangle.\ \mathsf{Seller} \to \mathsf{Buyer} : \mathsf{Fail}\langle \mathsf{reason} \rangle\ )$
6. $\qquad\qquad\qquad\qquad + $
7. $\quad \mathsf{Buyer} \to \mathsf{Seller} : \mathsf{Reject}\langle\rangle\ )$

Line 1. denotes the quote request and reply. Lines 2. and 7. are computational branches corresponding to acceptance and rejection of the quote respectively. If the quote is accepted, Buyer will send its credit card in the box $[\mathsf{quote}, \mathsf{card}]_{\mathsf{Buyer},\mathsf{Bank}}$ meaning that Seller cannot see it. The box is then forwarded to Bank together with the quote offered by Seller who checks that everything is fine (line 2.). If the transaction can be finalised, a receipt is forwarded to Buyer. Otherwise, a NotOK message will be delivered. Bank boxes the receipt so that it cannot be seen or changed by Seller.

4.1 Abstract Strand Semantics

The *abstract strand* semantics (AS semantics) is the minimum function $\{\!\{-\}\!\} : C \to 2^S \times (\mathscr{R} \to 2^S)$ (for S a set of strands) satisfying the rules in Table 1. The function inputs a choreography and returns a set of strands paired with a function that maps strands into a role ρ in $\mathscr{R}$ (all the possible runs for ρ). These strands are templates, and we may use substitutions to "plug in" alternate values for the parameters in the choreography. Since these parameters do not include the labels op_i, we define:

$$\text{(AS-Com)}\quad \frac{\{\!\{C_i\}\!\} = (S_i, \mathsf{who}_i)}{\{\!\{\Sigma_i \rho_1 \to \rho_2 : \mathsf{op_i}\langle \tilde{M_i}\rangle.\ C_i\}\!\} = \bigcup_i \left(\ \mathsf{extend}(S_i, [(\mathsf{op_i}, \tilde{M_i})]_{\rho_1\rho_2}, \rho_1, \rho_2, \mathsf{who}_i)\ \right)}$$

$$\text{(AS-Zero)}\quad \frac{}{\{\!\{\mathbf{0}\}\!\} = (\{+\bullet^{\rho}\}_{\rho}, \lambda\rho.\ \{+\bullet^{\rho}\})}$$

Table 1: Abstract Strand Semantics for Choreography

A substitution σ is a *parameter substitution* if for every label op_i, $\sigma(\mathsf{op}_i) = \mathsf{op}_i$. The *strand space of a choreography C* is the strand space generated by applying parameter substitutions to $\{\!\{C\}\!\}$. We say that a skeleton $\mathbb{A}$ is *over* $\{\!\{C\}\!\}$ if all of its strands belong to this strand space.

Rule (AS-Zero) gives semantics to the inactive choreography $\mathbf{0}$ by creating a strand $+\bullet^{\rho}$ for each role $\rho \in \mathscr{R}$. Rule (AS-Com) gives the semantics to the term (interaction) of a choreography. The idea is to prefix, for every branch, every strand of ρ_1 with $+[(\mathsf{op_i}, \tilde{M_i})]_{\rho_1\rho_2}$ and every strand of ρ_2 with $-[(\mathsf{op_i}, \tilde{M_i})]_{\rho_1\rho_2}$ where, in general, $(\mathsf{op}, \tilde{M})$ denotes the vector $(\mathsf{op}, M_1, \ldots, M_k)$. The main part is played by the function extend hereby defined as:

$$\mathsf{extend}(S, M, \rho_1, \rho_2, \mathsf{who}) = \left(\begin{array}{ll} S \backslash (\mathsf{who}(\rho_1) \cup \mathsf{who}(\rho_2)) & \cup \\ \left\{ a \Rightarrow s \ \mid \begin{array}{ll} (s \in \mathsf{who}(\rho_1) \wedge a = +M) & \vee \\ (s \in \mathsf{who}(\rho_2) \wedge a = -M) & \end{array} \right\} & \end{array} \right)$$

The above definition says that we include all those strands which are not in $\mathsf{who}(\rho_1)$ and in $\mathsf{who}(\rho_2)$. Then, we must prefix all those strands in $\mathsf{who}(\rho_1)$ with node $+M$ and all those strands in $\mathsf{who}(\rho_2)$ with $-M$. For well-formed choreographies, we have the following:

Proposition 5. *Let C be a well-formed choreography and* (S, who) *its semantics. Then each message* $[\tilde{M}]_{\rho_1\rho_2}$ *always originates in* $\mathsf{who}(\rho_1)$ *and can only be opened in* $\mathsf{who}(\rho_2)$.

Example 4 (Semantics of the Buyer-Seller Protocol)**.** Unlike in [3], because of the presence of corrupted roles (and participants), we cannot give the semantics of a choreography describing a security protocol simply by giving a set of executions. Therefore, the semantics of the buyer-seller protocol is a set of strands from which we would like to build the possible executions depending on which roles are compromised. Given the choreography in Example 3, we get the following strands:

$a)$ $+[(\mathsf{Req}, \mathsf{prod})]_{\mathsf{BS}} \Rightarrow -[(\mathsf{Reply}, \mathtt{quote})]_{\mathsf{SB}} \Rightarrow +[(\mathsf{Accept}, [(\mathtt{quote}, \mathtt{card})]_{\mathsf{BBk}})]_{\mathsf{BS}} \Rightarrow -[(\mathsf{Succ}, [\mathtt{receipt}]_{\mathsf{BkB}})]_{\mathsf{SB}}$

$b)$ $+[(\mathsf{Req}, \mathsf{prod})]_{\mathsf{BS}} \Rightarrow -[(\mathsf{Reply}, \mathtt{quote})]_{\mathsf{SB}} \Rightarrow +[(\mathsf{Accept}, [(\mathtt{quote}, \mathtt{card})]_{\mathsf{BBk}})]_{\mathsf{BS}} \Rightarrow -[(\mathsf{Fail}, \mathtt{reason})]_{\mathsf{SB}}$

$c)$ $+[(\mathsf{Req}, \mathsf{prod})]_{\mathsf{BS}} \Rightarrow -[(\mathsf{Reply}, \mathtt{quote})]_{\mathsf{SB}} \Rightarrow +[\mathsf{Reject}]_{\mathsf{BS}}$

$d)$ $-[(\mathsf{Req}, \mathsf{prod})]_{\mathsf{BS}} \Rightarrow +[(\mathsf{Reply}, \mathtt{quote})]_{\mathsf{SB}} \Rightarrow -[(\mathsf{Accept}, [(\mathtt{quote}, \mathtt{card})]_{\mathsf{BBk}})]_{\mathsf{BS}} \Rightarrow$
$\Rightarrow +[(\mathsf{Pay}, \mathtt{quote}, [(\mathtt{quote}, \mathtt{card})]_{\mathsf{BBk}})]_{\mathsf{SBk}} \Rightarrow -[(\mathsf{Ok}, [\mathtt{receipt}]_{\mathsf{BkB}})]_{\mathsf{BkS}} \Rightarrow +[(\mathsf{Succ}, [\mathtt{receipt}]_{\mathsf{BkB}})]_{\mathsf{SB}}$

$e)$ $-[(\mathsf{Req}, \mathsf{prod})]_{\mathsf{BS}} \Rightarrow +[(\mathsf{Reply}, \mathtt{quote})]_{\mathsf{SB}} \Rightarrow -[(\mathsf{Accept}, [(\mathtt{quote}, \mathtt{card})]_{\mathsf{BBk}})]_{\mathsf{BS}} \Rightarrow$
$\Rightarrow +[(\mathsf{Pay}, \mathtt{quote}, [(\mathtt{quote}, \mathtt{card})]_{\mathsf{BBk}})]_{\mathsf{SBk}} \Rightarrow -[(\mathsf{NotOk}, \mathtt{reason})]_{\mathsf{BkS}} \Rightarrow +[(\mathsf{Fail}, \mathtt{reason})]_{\mathsf{SB}}$

$f)$ $-[(\mathsf{Req}, \mathsf{prod})]_{\mathsf{BS}} \Rightarrow +[(\mathsf{Reply}, \mathtt{quote})]_{\mathsf{SB}} \Rightarrow -[\mathsf{Reject}]_{\mathsf{BS}}$

$g)$ $-[(\mathsf{Pay}, \mathtt{quote}, [(\mathtt{quote}, \mathtt{card})]_{\mathsf{BBk}})]_{\mathsf{SBk}} \Rightarrow +[(\mathsf{Ok}, [\mathtt{receipt}]_{\mathsf{BkB}})]_{\mathsf{BkS}}$

$h)$ $-[(\mathsf{Pay}, \mathtt{quote}, [(\mathtt{quote}, \mathtt{card})]_{\mathsf{BBk}})]_{\mathsf{SBk}} \Rightarrow +[(\mathsf{NotOk}, \mathtt{reason})]_{\mathsf{BkS}}$

where B is the buyer, S is the seller and Bk is the bank. Above, strands $a)$, $b)$ and $c)$ belong to B while $d)$, $e)$ and $f)$ belong to S. Strands $g)$ and $h)$ are instead the local behaviour of Bk.

4.2 Realized Skeletons for Choreography

We now apply the theory developed in the previous section to abstract spaces which are in fact the semantics of a choreography.

In the sequel we say that $\mathbb{A}$ is *over* $\{\!|C|\!\}$ whenever it is obtained from the regular, non compromised strands in $\{\!|C|\!\}$. The following result states that whenever (A1) is not applicable, we have reached a realized skeleton.

Lemma 1 (Realized Skeletons)**.** *Let C be a well-formed choreography and let $\mathbb{A}$ be a skeleton over $\{\!|C|\!\}$ such that (*A1*) is not applicable. Then $\mathbb{A}$ is realized.*

Proof. By Proposition 1, $\mathbb{A}'$ is realized if and only if all its cuts are solved. Let us assume, by contradiction, that $\mathsf{Cut}([M]_{\rho_1\rho_2}, B, \mathbb{A})$ is unsolved for some $[M]_{\rho_1\rho_2}$ and B.

By Proposition 2, we know that also $\mathsf{Cut}([M]_{\rho_1\rho_2}, B', \mathbb{A})$ is unsolved for $B' = \{b \mid n' \prec_{\mathbb{A}} n \text{ s.t. } [\tilde{M}]_{\rho_1\rho_2} \sqsubset b \sqsubseteq \mathsf{msg}(n') \wedge \mathsf{rcv}(b) \notin R\}$ for some $\preceq_{\mathbb{A}}$-minimal input node n in $\mathsf{Cut}([M]_{\rho_1\rho_2}, B, \mathbb{A})$ and $\rho_1 \notin R$. As a consequence, we also have that $[\tilde{M}]_{\rho_1\rho_2} \dagger^{B'} \mathsf{msg}(n)$.

Now, if we prove the existence of some positive node $m \notin \mathbb{A}$ such that $\forall m' . m' \prec_{\mathbb{A}} n \vee m' \Rightarrow^{+} m$ implies $[\tilde{M}]_{\rho_1\rho_2} \odot^{B'} \mathsf{msg}(m')$ where $[\tilde{M}]_{\rho_1\rho_2} \dagger^{B'} \mathsf{msg}(m)$ and $m \notin R$ then we can apply (A1) to $\mathbb{A}$ hence having a contradiction. We distinguish two cases:

- $B' = \emptyset$. In this case, the unsolved cut is saying that we must explain where the box c has been created. As ρ_1 is not compromised, we must add a node belonging to ρ_1 sending c. The existence of such a node is ensured by well-formedness.
- $B' \neq \emptyset$. As B is non-empty, then we must explain how c has come out of some message box $[\tilde{M'}]_{\rho_3\rho_4}$ in B. But if that is the case, as ρ_4 is not compromised, a node belonging to ρ_4 must have performed such operation. The existence of such a node is ensured by well-formedness.

Note that in both cases above, we are exploiting the fact that the two well-formedness conditions impose that the operations for creation and opening of a box are performed consistently on the same choreography branches i.e. role strands. □

The following result states that whenever (A2) is not applicable to $\mathbb{A}$ then $\mathbb{A}$ is DG.

Lemma 2. *Let C be a well-formed choreography and let $\mathbb{A}$ be a skeleton over $\{\!|C|\!\}$ such that (*A2*) is not applicable. Then $\mathbb{A}$ is DG.*

Proof. If that is not the case then, by definition of delivery guaranteed skeleton, we would be able to apply (A2). This is simply because whenever we add a positive node n to $\mathbb{A}$ we always have another strand belonging to a different role and containing a negative node m such that $\mathsf{msg}(m) = \mathsf{msg}(n)$. □

We finally have the following two results:

Theorem 2 (Soundness)**.** *Let $\mathbb{A}$ be a single-stranded skeleton over $\{\!|C|\!\}$ and let $\mathbb{A} \longrightarrow_S^* \mathbb{A}' \not\rightarrow$. Then, $\mathbb{A}'$ is a DG shape.*

Proof. By the previous lemmas, we know that $\mathbb{A}'$ is realized and DG. We must prove that there exists a homomorphism $H : \mathbb{A} \mapsto \mathbb{A}'$ which is a shape.

As $\mathbb{A} \longrightarrow_S^* \mathbb{A}'$ then, by Proposition 4, we can choose $H = H_k \circ \ldots \circ H_0$ where $H_i : \mathbb{A}_i \mapsto \mathbb{A}_{i+1}$ for for $\mathbb{A}_0 = \mathbb{A}$ and $\mathbb{A}_{k+1} = \mathbb{A}'$ and some $\mathbb{A}_i$. We shall prove that $H_k \circ \ldots \circ H_i : \mathbb{A}_i \mapsto \mathbb{A}'$ is a shape for $\mathbb{A}_i$ for all i. We do it by induction on $j = k - i$.

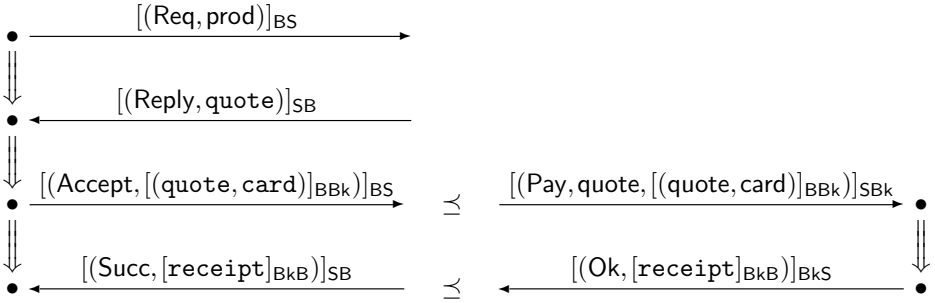
- **Base Case.** $j = 1$. We have to prove that $H_k : \mathbb{A}_k \mapsto \mathbb{A}'$ is a shape for $\mathbb{A}_k$. By Proposition 3, we know that $\mathbb{A}_k$ is not realized and/or not DG. Hence, H_k must be the minimum homomorphism mapping $\mathbb{A}_k$ to a DG realized skeleton. In fact, both (A1) and (A2), add the minimum node explaining a box or receiving a pending output.
- **Inductive Case.** Let us assume that $j = i+1$. By induction hypothesis we know that $H_k \circ \ldots \circ H_{i+1} : \mathbb{A}_{i+1} \mapsto \mathbb{A}'$ is a shape for $\mathbb{A}_{i+1}$. But then, as augmentations are minimal strictly monotone embedding with respect to shapes, we have that also $H_k \circ \ldots \circ H_i : \mathbb{A}_i \mapsto \mathbb{A}'$ is a shape for $\mathbb{A}_i$.

□

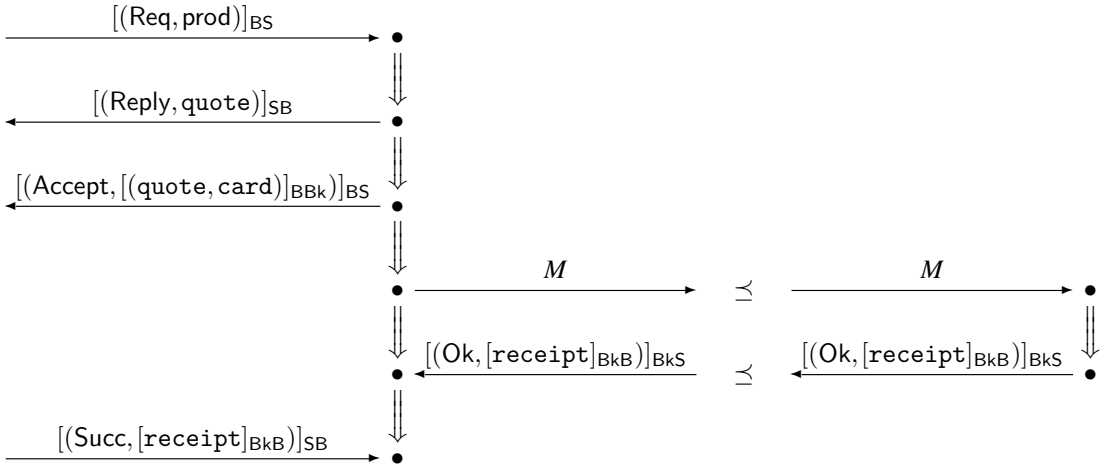
Theorem 3 (Termination)**.** *Let $\mathbb{A}$ be a single-stranded skeleton over $\{\!\{C\}\!\}$. Then, we can reduce $\mathbb{A}$ only a finite number of times.*

Proof. $\{\!\{C\}\!\}$ is finite and the reduction rules are augmentation (increase the number of nodes). As the same node cannot be added twice, we must eventually exhaust all nodes. □

Example 5 (Shapes of the Buyer-Seller Protocol)**.** We show how to compute some shapes of the Buyer-Seller protocol starting from its semantics given in the previous section. We start from the buyer's strand a) assuming that seller is compromised. Applying (A1) to its fourth node, we get:



Note that, we have actually applied (A1) twice: the second time it was applied to the first node of the new strand and its result was only adding the relation $\preceq$. The image of the shape for strand b), the case when the bank does not accept the transaction, is similar. Let us now consider d) and let us assume that buyer is compromised. In this case, for $M = [(\mathsf{Pay}, \mathtt{quote}[(\mathtt{quote}, \mathtt{card})]_{\mathsf{BBk}})]_{\mathsf{SBk}}$, by applying (A1) (twice) we get:



Example 6. Let us consider a slightly different version of the Buyer-Seller protocol, where the buyer does not include the quote together with her credit card. In particular we would have the new following strands (the missing ones are unchanged):

$$a') \quad +[(\mathsf{Req},\mathsf{prod})]_{\mathsf{BS}} \Rightarrow -[(\mathsf{Reply},\mathtt{quote})]_{\mathsf{SB}} \Rightarrow +[(\mathsf{Accept},[\mathtt{card}]_{\mathsf{BBk}})]_{\mathsf{BS}} \Rightarrow -[(\mathsf{Succ},[\mathtt{receipt}]_{\mathsf{BkB}})]_{\mathsf{SB}}$$

$$b') \quad +[(\mathsf{Req},\mathsf{prod})]_{\mathsf{BS}} \Rightarrow -[(\mathsf{Reply},\mathtt{quote})]_{\mathsf{SB}} \Rightarrow +[(\mathsf{Accept},[\mathtt{card}]_{\mathsf{BBk}})]_{\mathsf{BS}} \Rightarrow -[(\mathsf{Fail},\mathtt{reason})]_{\mathsf{SB}}$$

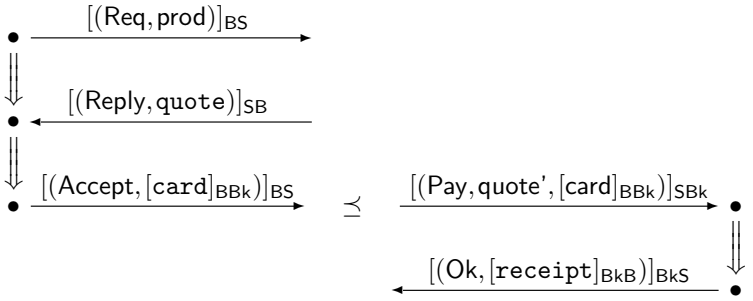
$$d') \quad -[(\mathsf{Req},\mathsf{prod})]_{\mathsf{BS}} \Rightarrow +[(\mathsf{Reply},\mathtt{quote})]_{\mathsf{SB}} \Rightarrow -[(\mathsf{Accept},[\mathtt{card}]_{\mathsf{BBk}})]_{\mathsf{BS}} \Rightarrow$$
$$\Rightarrow +[(\mathsf{Pay},\mathtt{quote},[\mathtt{card}]_{\mathsf{BBk}})]_{\mathsf{SBk}} \Rightarrow -[(\mathsf{Ok},[\mathtt{receipt}]_{\mathsf{BkB}})]_{\mathsf{BkS}} \Rightarrow +[(\mathsf{Succ},[\mathtt{receipt}]_{\mathsf{BkB}})]_{\mathsf{SB}}$$

$$e') \quad -[(\mathsf{Req},\mathsf{prod})]_{\mathsf{BS}} \Rightarrow +[(\mathsf{Reply},\mathtt{quote})]_{\mathsf{SB}} \Rightarrow -[(\mathsf{Accept},[\mathtt{card}]_{\mathsf{BBk}})]_{\mathsf{BS}} \Rightarrow$$
$$\Rightarrow +[(\mathsf{Pay},\mathtt{quote},[\mathtt{card}]_{\mathsf{BBk}})]_{\mathsf{SBk}} \Rightarrow -[(\mathsf{NotOk},\mathtt{reason})]_{\mathsf{BkS}} \Rightarrow +[(\mathsf{Fail},\mathtt{reason})]_{\mathsf{SB}}$$

$$g') \quad -[(\mathsf{Pay},\mathtt{quote},[\mathtt{card}]_{\mathsf{BBk}})]_{\mathsf{SBk}} \Rightarrow +[(\mathsf{Ok},[\mathtt{receipt}]_{\mathsf{BkB}})]_{\mathsf{BkS}}$$

$$h') \quad -[(\mathsf{Pay},\mathtt{quote},[\mathtt{card}]_{\mathsf{BBk}})]_{\mathsf{SBk}} \Rightarrow +[(\mathsf{NotOk},\mathtt{reason})]_{\mathsf{BkS}}$$

If the seller is corrupted, starting from g') and applying (A1) to its first node, we get the realized skeleton:



The realized skeleton above shows a flaw, or at least an undesirable aspect of this version of the protocol. The value quote that the client accepted can be different from quote' received by the bank, allowing for the seller to cheat on the quote agreed with the buyer.

5 Conclusions

In this paper, we have used the strand space framework to study the possible behaviors of choreographies executing in the presence of compromised principals. In this framework, the strands of the uncompromised regular participants can freely interact with each other and with behaviors possible for corrupted parties. We clarified these behaviors by presenting a pair of transition rules which generate all of the minimal, essentially different executions.

It is a strength of this approach that it allows us to formulate and characterize a number of interesting properties. For instance, what about the relationship between shapes (namely minimal executions) and other, possibly non-minimal executions? One might expect that non-minimal executions would be disjoint unions of copies of shapes. However, this intuition requires a property of choreographies, which may be characterized syntactically. In effect, it requires that when the choreography has a choice, then the same principals are active across both branches of the choice (except possibly the last principal on one branch). This corresponds to an assumption of [6]. We also conjecture that, under these assumptions, shapes are *run-once* i.e. they are such that there is at most one strand belonging to each role. In future work we intend to explore properties of this kind, in particular when the choreography language is extended with parallel composition and recursive behaviour.

We also intend to study the relation between protocol descriptions at the choreography-and-box level and at the concrete cryptographic level. We intend to investigate properties of protocol transformations in general [10] in order to develop fine-grained principles governing how to generate cryptographic implementations for choreographies requiring security infrastructures.

References

[1] Michael Backes, Agostino Cortesi, Riccardo Focardi, and Matteo Maffei. A calculus of challenges and responses. In *FMSE '07: Proceedings of the 2007 ACM workshop on Formal methods in security engineering*, pages 51–60, New York, NY, USA, 2007. ACM.

[2] Lorenzo Bettini, Mario Coppo, Loris D'Antoni, Marco De Luca, Mariangiola Dezani-Ciancaglini, and Nobuko Yoshida. Global progress in dynamically interleaved multiparty sessions. In *19th International Conference on Concurrency Theory (Concur'08)*, LNCS, pages 418–433. Springer, 2008.

[3] Marco Carbone and Joshua Guttman. Execution models for choreographies and cryptoprotocols. In *Pre-proceedings of PLACES'09*, 2009.

[4] Marco Carbone, Kohei Honda, and Nobuko Yoshida. Structured Communication-Centred Programming for Web Services. In *16th European Symposium on Programming (ESOP'07)*, volume 4421 of *LNCS*, pages 2–17. Springer, 2007.

[5] Ricardo Corin, Pierre-Malo Deniélou, Cédric Fournet, Karthikeyan Bhargavan, and James J. Leifer. A secure compiler for session abstractions. *Journal of Computer Security*, 16(5):573–636, 2008.

[6] Ricardo Corin, Pierre-Malo Deniélou, Cédric Fournet, Karthikeyan Bhargavan, and James J. Leifer. Cryptographic protocol synthesis and verification for multiparty sessions. In *Proceedings, IEEE Symposium on Computer Security Foundations*. IEEE Computer Society Press, July 2009.

[7] T. Dierks and C. Allen. The TLS protocol. RFC 2246, January 1999.

[8] S.F. Doghmi, Joshua Guttman, and Javier Thayer. Searching for shapes in cryptographic protocols. In *Proc. TACAS'07*, volume 4424, pages 523–537, 2007.

[9] Joshua Guttman. Cryptographic protocol composition via the authentication. In *Proc. FOSSACS'09*, volume 5504 of *LNCS*, pages 303–317, 2009.

[10] Joshua D. Guttman. Transformations between cryptographic protocols. In P. Degano and L. Viganò, editors, *Automated Reasoning in Security Protocol Analysis, and Workshop on Issues in the Theory of Security (ARSPA-WITS)*, LNCS. Springer, 2009.

[11] Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo. Language Primitives and Type Disciplines for Structured Communication-based Programming. In *7th European Symposium on Programming (ESOP'98)*, volume 1381 of *LNCS*, pages 22–138. Springer-Verlag, 1998.

[12] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. In *35th Symposium on Principles of Programming Languages (POPL'08)*, pages 273–284. ACM, 2008.

[13] Dimitris Mostrous, Nobuko Yoshida, and Kohei Honda. Global principal typing in partially commutative asynchronous sessions. In *ESOP Proceedings*, LNCS. Springer, March 2009.

[14] Kaku Takeuchi, Kohei Honda, and Makoto Kubo. An Interaction-based Language and its Typing System. In *PARLE'94*, volume 817 of *LNCS*, pages 398–413. Springer-Verlag, 1994.

[15] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1), 1999.