Execution Models for Choreographies and Cryptoprotocols

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

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

Abstract

A choreography describes a transaction in which several principals interact. Since choreographies frequently describe business processes affecting substantial assets, we need a security infrastructure in order to implement them safely. As part of a line of work devoted to generating cryptoprotocols from choreographies, we focus here on the execution models suited to the two levels.

We give a strand-style semantics for choreographies, and propose a special execution model in which choreography-level messages are faithfully delivered exactly once. We adapt this model to handle multiparty protocols in which some participants may be compromised.

At level of cryptoprotocols, we use the standard Dolev-Yao execution model, with one alteration. Since many implementations use a ”nonce cache” to discard multiply delivered messages, we provide a semantics for at-most-once delivery.

1 Introduction

Choreographies are global descriptions of transactions including business or financial transactions. They describe the intertwined behavior of several principals as they negotiate some agreement and—frequently—commit some state change. A key idea is *end-point projection* [5], which converts a global description into a set of descriptions that determine the local behavior of the individual participants in a choreography. Conversely, *global synthesis* of a choreography from local behaviors is also sometimes possible, i.e. meshing a set of local behaviors into a comprehensive global description [11].

Because these transactions may transfer sums of money and other objects of value, or may communicate sensitive information among the principals, they require a security infrastructure. It would be desirable to synthesize a cryptographic protocol directly from a choreography description, to control how adversaries can interfere with transactions among compliant principals. Corin et al. [6] have made a substantial start on this problem, with further advances described in [3]. However, many questions remain, for instance how to optimize the generated cryptographic protocols, how best to establish that they are always correct, and indeed how best to define their correctness.

This last question concerns how to state what control the protocol should provide, against adversaries trying to interfere with transactions. It is a substantial question because the execution model that choreographies use is quite distant from the execution model cryptographic protocols are designed to cope with. For instance, choreographies use an execution model—or communication model—in which messages are never received by any party other than the intended recipient, or if the formalism represents channels, they are received only over the channel. Moreover, messages are always delivered if the recipient is willing to receive the message. Messages are delivered only if they were sent, and specifically only if they were sent by the expected peer. Finally, they are delivered only once. These aspects of the model mean that confidentiality and integrity properties are built into the underlying assumptions. A security infrastructure is intended to justify exactly these assumptions, i.e. to provide a set of behaviors in which these assumptions are satisfied.

Naturally, these behaviors must be achieved within an underlying model in which the adversary is much stronger. In this model—typically called the *Dolev-Yao model*, after a paper [8] in which Dolev and Yao formalized ideas suggested by Needham and Schroeder [12]—all messages may be received by the adversary, so that confidentiality needs to be achieved by encryption. They may be delivered zero times,

[*]The author was supported by EPSRC grant EP/F002114

once, or repeatedly, and they may be misdelivered to the wrong participant. When delivered, a message may appear to come from a participant that did not send it. The adversary may alter messages in transit, including applying cryptographic operations using keys that he knows, or may obtain by manipulating the protocol.

Digital signatures may be used to notify a recipient reliably of the source of a message (and of the integrity of its contents). Symmetric encryption may also be used to ensure authenticity: a recipient knows that the encrypted message was prepared by a party that knew the secret key, and intended it for a peer that also knew the secret key. Nonces, which are simply randomly chosen bitstrings, may be used to ensure freshness. The principal P that chose a nonce knows, when receiving a message containing it, that the nonce was inserted after P chose it. Moreover, if P engages in many sessions and associates a different nonce with each, P can ensure that messages containing one nonce cannot be misdirected to a session using a different nonce.

In this paper, we begin the process of relating the Dolev-Yao model of execution to the choreography execution model. This is a key step in generating cryptographic protocols and proving them faithful to the intent of the choreography. In particular, we represent the two execution models using the strand space model [13, 10].

Goals of this Paper. We provide a few definitions and an example to indicate how the strand space framework can relate choreographies to the cryptographic protocols that implement them.

In particular, we consider a very simple choreography language, and provide a semantics for it as a set of "abstract bundles." That is, each session of the protocol executes according to one of the bundles predicted by the semantics. Moreover, any collection of sessions that may have occurred takes the following form: its events partition into bundles that are obtained by instantiating the parameters in bundles given in the semantics. Also, if two nodes belong to different partition elements, there is no $\preceq$ ordering between them, unless the executions are generated as parts of some higher-level choreography that might determine a causal ordering.

We call this an *abstract bundle semantics* because it builds in the assumptions of the choreography level: messages do not have explicit cryptographic operations, and the choreography-level communication assumptions are satisfied. Messages are always delivered exactly once; sender and recipient are never mismatched; no message is created by adversary operations. We must connect this idealized semantics with a more realistic semantics at the cryptographic level, in which the adversary may be active.

One peculiarity of our message datatype is that we allow "boxes." A box $[\tilde{M}]_{\rho\rho'}$ is a message prepared on role ρ that can be opened only by a principal playing role ρ'. At the choreography level, this property is enforced by a type system. We use these boxes to make explicit the confidentiality and authentication requirements of a choreography in the case where some roles are played by compromised participants. However, in this article, we focus on the simplest case, in which no participants are compromised. That is, we will assume here, that any participant who is sent a box, will behave only as predicted by the choreography.

Our semantics at the *cryptographic level* is a standard strand space treatment, except for one ingredient. Namely, this semantics assumes that some kinds of messages are delivered at most once. These are session-initiating messages that contain a nonce, or in some protocols a freshly generated session key. Implementations now use a nonce-caching technique in which the nonces of previously executed sessions are retained in a cache. A new incoming message contains a nonce which is compared against the cache; if it is present, then with overwhelming probability there has been a replay attempt, and the message is discarded. Otherwise, the nonce is recorded and the session proceeds. So as not to need to retain nonces forever, implementations typically combine this with a timestamp, and assume that uncompromised principals are loosely synchronized. A message with too old a timestamp is discarded.

Nonces may be dropped from the cache when their timestamps have expired. In this approach, the nonce and the timestamp must appear digitally signed in the incoming message to prevent manipulation by the adversary.

We define a cryptographic protocol to properly implement a choreography if, when abstracting its possible executions in this at-most-once semantics, we obtain exactly the possible executions of the abstract bundle semantics for the choreography.

We explore here a simple example in which the participants are well-known to each other from the start of the transaction. However, the ideas also apply when additional participants may be chosen during execution, and keys must be distributed as part of the message flow.

2 Strand Spaces

Strand spaces [13, 10] were developed as a simplest possible model for cryptographic protocol analysis, but are also applicable to other kinds of distributed systems. In strand spaces, we consider *strands*, behavioural traces for roles represented as finite linear sequences of transmission and reception events. The model provides techniques for analysing how various strands can be combined together in a run of a protocol including some adversary behaviour. Formally, for A being a generic set of messages,

Definition 1 (Strand Space). *A* directed term *is a pair denoted by $\pm a$ (for $a \in A$) where $\pm \in \{-,+\}$ is a direction with $+$ representing transmission and $-$ reception. A trace is an element of $(\pm A)^*$, the set of infinite sequences of directed terms.*
A strand space *is a set S equipped with a trace mapping $\mathtt{tr} : S \to (\pm A)^*$ and its elements are called* strands.

In the sequel, if s is a strand in some strand space S 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 following 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.

But how can strands be combined together in order to represent executions of a protocol? This is precisely captured by the notion of *bundle*:

Definition 2 (Bundle). *Given a strand space S, a finite acyclic directed graph $\mathscr{B} = (\mathscr{N}, \mathscr{E}, \preceq_{\mathscr{B}})$ is a* bundle *if*

1. *$\mathscr{N}$ is a set of strand nodes in S such that if $n \in \mathscr{N}$ and $m \Rightarrow n$, then $m \in \mathscr{N}$;*
2. *$\mathscr{E} = \rightarrow_{\mathscr{B}} \cup \Rightarrow_{\mathscr{B}}$ where*
   - *(a) $\Rightarrow_{\mathscr{B}}$ is the restriction of $\Rightarrow$ to nodes in $\mathscr{N}$; and*
   - *(b) for any reception node $n \in \mathscr{N}$, there is exactly one transmission node $m \in \mathscr{N}$ such that $m \rightarrow_{\mathscr{B}} n$ and $\mathsf{msg}(m) = \mathsf{msg}(n)$;*
3. *$n \preceq_{\mathscr{B}} m$ iff there is a path from n to m in $\mathscr{B}$.*

A *bundle* is a causally well-founded graph – essentially, a Lamport diagram – built from strands and transmission edges. A transmission edge $m \to n$ is possible when m is a transmission node; n is a reception node; and the message transmitted on m is the same as the message received on n. Note that the relation $\preceq_{\mathscr{B}}$ is a well-founded partial order, meaning that the *bundle induction* principle holds, that every non-empty set of nodes of $\mathscr{B}$ contains $\preceq_{\mathscr{B}}$-minimal members.

The notions of strand and bundle, and the principle of bundle induction, are the essential ingredients in the strand space model. Choices – such as what operations the adversary strands offer, or what additional closure properties bundles may satisfy – can vary to model different problems concerning cryptographic protocols or distributed communication more generally.

Example. We briefly introduce an example in order to clarify the concepts introduced above. Let S be composed by the following strands:

(1) $n_1 \Rightarrow n_2$ (2) $n_3 \Rightarrow n_4$ (3) $n_5 \Rightarrow n_6$ (4) $n_7 \Rightarrow n_8 \Rightarrow n_9 \Rightarrow n_{10}$ (5) $n_{11} \Rightarrow n_{12}$

where

$\mathsf{msg}(n_1) = +$”Hello”	$\mathsf{msg}(n_2) = -$”Bye”		
$\mathsf{msg}(n_3) = +$”Good luck”	$\mathsf{msg}(n_4) = -$”Thanks”		
$\mathsf{msg}(n_5) = -$”Good luck”	$\mathsf{msg}(n_6) = +$”Thanks”		
$\mathsf{msg}(n_7) = -$”Hello”	$\mathsf{msg}(n_8) = -$”Good luck”	$\mathsf{msg}(n_9) = +$”Thanks”	$\mathsf{msg}(n_{10}) = +$”Bye”
$\mathsf{msg}(n_{11}) = -$”Thanks”	$\mathsf{msg}(n_{11}) = +$”Bye”		

Below, we report two possible executions in the strand space S (for clarity, we label $\rightarrow$ with the corresponding message):

$$\begin{tikzcd}
n_3 \arrow[r, "\text{Good luck}"] \arrow[d, Rightarrow] & n_5 \arrow[d, Rightarrow] \\
n_4 & n_6 \arrow[l, "\text{Thanks}"']
\end{tikzcd}
\qquad
\begin{tikzcd}
n_1 \arrow[r, "\text{”Hello”}"] \arrow[ddd, Rightarrow] & n_7 \arrow[d, Rightarrow] & \\
 & n_8 \arrow[d, Rightarrow] & n_3 \arrow[l, "\text{”Good Luck”}"'] \arrow[d, Rightarrow] \\
 & n_9 \arrow[r, "\text{”Thanks”}"] \arrow[d, Rightarrow] & n_4 \\
n_2 & n_{10} \arrow[l, "\text{”Bye”}"'] &
\end{tikzcd}$$

Note that strand (5) could interfere allowing for the following bundle:

$$\begin{tikzcd}
n_1 \arrow[r, "\text{”Hello”}"] \arrow[ddd, Rightarrow] & n_7 \arrow[d, Rightarrow] & & \\
 & n_8 \arrow[d, Rightarrow] & n_3 \arrow[l, "\text{”Good Luck”}"'] & \\
 & n_9 \arrow[rr, "\text{”Thanks”}"] & & n_{11} \arrow[d, Rightarrow] \\
n_2 & & & n_{12} \arrow[lll, "\text{”Bye”}"']
\end{tikzcd}$$

3 An Execution Model for Choreography

3.1 The Calculus

Syntax. Let ρ range over the set of roles $\mathscr{R}$. The syntax of our choreography mini-language (based on the Global Calculus [5]) is given by the following grammar:

$$C ::= \quad \Sigma_i\, \rho_1 \rightarrow \rho_2 : \mathsf{op_i}\langle \tilde{M}_i \rangle.\ C_i \mid \quad \mathbf{0} \qquad\qquad M ::= \quad v \mid [\tilde{M}]_{\rho_1\rho_2}$$

Above, the term $\Sigma_i\, \rho_1 \rightarrow \rho_2 : \mathsf{op_i}\langle \tilde{M}_i \rangle.\ C_i$ 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. A message M can either be a value v or a box $[\tilde{M}]_{\rho_1\rho_2}$. The latter denotes a tuple of messages M_i from ρ_1 that can only be opened by ρ_2.

Syntactic Assumption. Let snd be a partial function such that $\mathsf{snd}(\Sigma_i \rho_1 \to \rho_2 : \mathsf{op_i}\langle \tilde{M_i}\rangle.\ C_i) = \rho_1$. Then, we assume that, for every choreography C:

- all op's are distinct.
- in any path in a choreography syntax tree, a box $[\tilde{M}]_{\rho_1\rho_2}$ has to occur first in an interaction whose sender is ρ_1 and can only be opened by ρ_2 in later interaction;
- if $C = \Sigma_i \rho_1 \to \rho_2 : \mathsf{op_i}\langle \tilde{M_i}\rangle.\ C_i$ then either $C_i = \mathbf{0}$ or $\mathsf{snd}(C_i) = \rho_2$ for all C_i;

The last assumption above requires that the receiving role in an interaction is always the transmitting role in the subsequent interaction. All the assumptions above can be statically checked [4].

LTS Semantics. Our mini-language can be equipped with a standard trace semantics with configurations $C \xrightarrow{\mu} C'$ where μ contains the parameters of the interaction performed i.e. it ranges over the set $\mathscr{R} \times \mathscr{R} \times \mathscr{O} \times \mathscr{M}$ where $\mathscr{O}$ is the set of operators op and $\mathscr{M}$ the set of messages. The following rule formally defines the relation $\xrightarrow{\mu}$ which is taken up to commutativity and associativity of $+$:

$$(\textsc{C-Com})\ \frac{}{\Sigma_i \rho_1 \to \rho_2 : \mathsf{op_i}\langle \tilde{M_i}\rangle.\ C_i \xrightarrow{(\rho_1,\rho_2,\mathsf{op_i},\tilde{M})} C_i}$$

Buyer-Seller Example. We report a variant of the Buyer-Seller financial protocol [5]. A buyer (or client) C asks a seller S for a quote about a product prod. If the quote is accepted, C will send its credit card card to S who will forward it to a bank B. The bank will check if the payment can be done and, if so, reply with a receipt receipt which S will forward to C. In our syntax:

$$\begin{array}{rlll}
1. & \multicolumn{3}{l}{\mathsf{C} \to \mathsf{S} : \mathtt{req}\langle \mathsf{prod}\rangle.\ \mathsf{S} \to \mathsf{C} : \mathtt{reply}\langle \mathsf{quote}\rangle.} \\
2. & (& \mathsf{C} \to \mathsf{S} : \mathtt{ok}\langle [\mathsf{card}]_{\mathsf{CB}}\rangle.\ \mathsf{S} \to \mathsf{B} : \mathtt{pay}\langle [\mathsf{card}]_{\mathsf{CB}}\rangle.\ (& \mathsf{B} \to \mathsf{S} : \mathtt{okcf}\langle [\mathsf{receipt}]_{\mathsf{BC}}\rangle. \\
3. & & & \mathsf{S} \to \mathsf{C} : \mathtt{rcpt}\langle [\mathsf{receipt}]_{\mathsf{BC}}\rangle \\
4. & & & \qquad\quad + \\
5. & & & \mathsf{B} \to \mathsf{S} : \mathtt{nopaycf}\langle\rangle. \\
6. & & & \mathsf{S} \to \mathsf{C} : \mathtt{nopay}\langle\rangle \quad) \\
7. & & \qquad\qquad + & \\
8. & & \mathsf{C} \to \mathsf{S} : \mathtt{refuse}\langle \mathsf{reason}\rangle) &
\end{array}$$

Line 1. denotes the quote request and reply. Lines 2. and 8. are computational branches corresponding to acceptance and rejection of the quote respectively. If the quote is accepted, C will send its credit card in the box $[\mathsf{card}]_{\mathsf{CB}}$ meaning that S cannot see it. The card number is then forwarded to B who can open the box (line 2.). If the transaction can be finalised a receipt is forwarded to C. Otherwise, a nopay notification will be sent. B boxes the receipt so that it cannot be seen or changed by S.

3.2 Abstract Bundle Semantics (ABS).

We introduce an alternative semantics for choreography based on bundles defined as judgements of the form:

$$\models C \rhd \{(\mathscr{B}_1, \mathsf{who}_1), \ldots, (\mathscr{B}_i, \mathsf{who}_i)\}$$

where $(\mathscr{B}, \mathsf{who})$ is a *bundle environment*. Given a role ρ, $\mathsf{who}(\rho)$ denotes the strand in the bundle $\mathscr{B}$ associated to the behaviour of ρ. The abstract bundle semantics $[\![C]\!] = \{(\mathscr{B}_1, \mathsf{who}_1), \ldots, (\mathscr{B}_i, \mathsf{who}_i)\}$ if

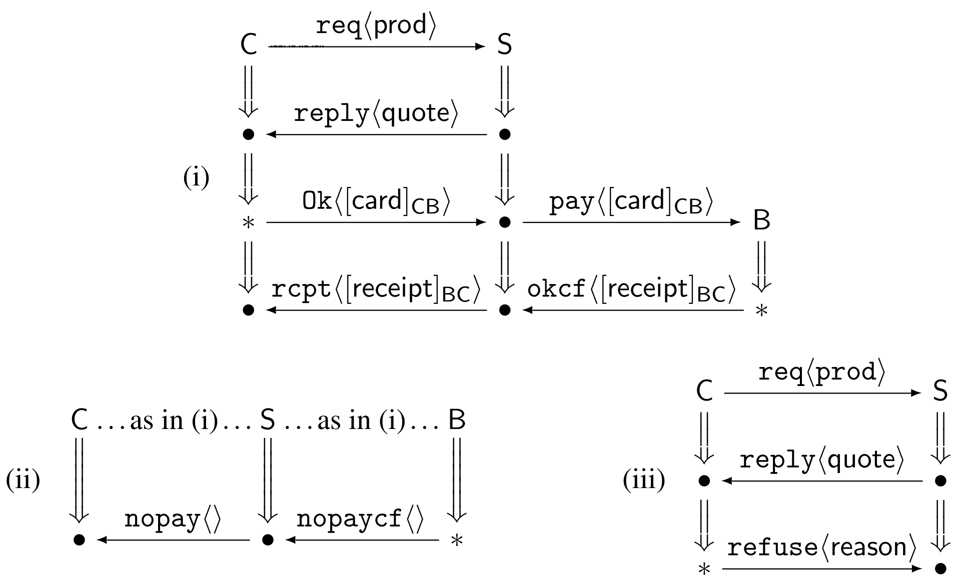


Figure 1: Bundles for the Buyer-Seller protocol

and only if $\models C \rhd \{(\mathscr{B}_1,\mathsf{who}_1),\ldots,(\mathscr{B}_i,\mathsf{who}_i)\}$. The relation $\models$ is the minimum relation satisfying the following:

$$(\text{ABS-COM})\qquad \frac{\forall i.\ \models C_i \rhd \{(\mathscr{B}_{i1},\mathsf{who}_{i1}),\ldots,(\mathscr{B}_{ij_i},\mathsf{who}_{ij_i})\}}{\models \Sigma_i\,\rho_1 \to \rho_2 : \mathsf{op_i}\langle\tilde{M}_i\rangle.\ C_i \rhd \left(\ \bigcup_i\{(\mathscr{B}_{ij_i},\mathsf{who}_{ij_i})\}_{j_i}[\rho_1,\rho_2,\mathsf{op}_i(\tilde{M}_i)]\ \right)}$$

$$(\text{ABS-ZERO})\qquad \frac{\mathsf{e}\text{ fresh}}{\emptyset \models \mathbf{0} \rhd (\{\mathsf{e}^{\rho}\}_{\rho},\lambda\rho.\ \mathsf{e}^{\rho})}$$

The abstract bundle semantics provides a set of bundles which represents all executions of the protocol described by the choreography. In (ABS-COM), $(\mathscr{B}_{ij_i},\mathsf{who}_{ij_i})[\rho_1,\rho_2,\mathsf{op}_i(\tilde{M}_i)]$ denotes a new bundle obtained from $\mathscr{B}_{ij_i}$ where the two strands $\mathsf{who}_{ij_i}(\rho_1)$ and $\mathsf{who}_{ij_i}(\rho_2)$ are prefixed with the events $+\mathsf{op}_i(\tilde{M}_i)$ and $-\mathsf{op}_i(\tilde{M}_i)$ respectively. The function who_{ij_i} is updated accordingly. Formally,

$$(\mathscr{B},\mathsf{who})[\mu] \;=\; (\quad (\mathscr{N}\cup\{n_i\}_i,\mathscr{E}\cup\{n_i \Rightarrow \mathsf{who}(\rho_i)\}_i\cup\{n_1\to n_2\},\preceq'),\quad \mathsf{who}[\rho_i\mapsto n_i\Rightarrow\mathsf{who}(\rho_i)]_i\quad)$$

where $\prec'$ is the update of $\prec_{\mathscr{B}}$ according to the new elements added to the bundle and $\mathscr{B}=(\mathscr{N},\mathscr{E},\preceq_{\mathscr{B}})$. The operation above is applied to all those bundles obtained from the semantics of each branch and the result will be their union. In (ABS-ZERO), we augment the set A with fresh events $\{\mathsf{e}^{\rho}\}\in E$ in order to distinguish each strand.

ABS Example. The ABS for the Buyer-Seller protocol has three bundles corresponding to its possible executions, namely: (i) C accepts the quote and B successfully finalises the transaction sending back a receipt; (ii) C accepts the quote but B does not accept the payment; and (iii) Buyer does not accept the quote with reason reason and the protocol terminates. The three corresponding bundles are reported in Fig. 1. The nodes marked with $*$ are those points where there is a possibility of branching i.e. bundle (ii) is identical to (i) up to its $*$ while (iii) is identical to (i) and (ii) up to its $*$. Note that (iii) only involves roles C and S.

In the sequel, let $(\mathscr{B},\mathsf{who})\backslash[\mu]$ be defined as follows:

$$(\mathscr{B},\mathsf{who})\backslash[\mu] \;=\; \begin{cases} \mathscr{B}' & \text{if}\quad \mathscr{B}=(\mathscr{B}',\mathsf{who})[\mu] \\ \text{undefined} & \text{otherwise} \end{cases}$$

Intuitively, the operation above is inverse to $(\mathscr{B},\mathsf{who})[\mu]$ i.e. removes the first communication from a bundle (if equal to μ, undefined otherwise). We can then conclude this section with a result that relates the LTS semantics to the bundle semantics.

Theorem 1. *Let C be a choreography. Then,*

1. *if $C \xrightarrow{\mu} C'$ then there exists a bundle $\mathscr{B}$ in $[\![C]\!]$ such that $[\![C']\!] = [\![C]\!]\backslash(\{\mathscr{B}\} \cup L) \cup \{\mathscr{B}\backslash[\mu]\}$ for $L = \{\mathscr{B}' \mid \mathscr{B} \in [\![C]\!] \wedge \mathscr{B}\backslash[\mu]$ is undefined$\}$;*
2. *if $\mathscr{B}\backslash[\mu]$ is defined and $\mathscr{B} \in [\![C]\!]$ then there exists C' such that $C \xrightarrow{\mu} C'$.*

4 An execution model for Cryptoprotocols

Cryptographic protocols are modelled by strand spaces where the set of messages a is more general. Formally, crypto-level messages, denoted by the syntactic category t have the following syntax:

$$t ::= \quad \tilde{v} \quad | \quad \{|\tilde{t}|\}_K$$

Above, the value v ranges over the disjoint union of infinite sets of nonces (denoted by N), atomic keys (denoted by K) and other basic values. We will write a sequence of messages in the form $v_1\hat{\ }\ldots\hat{\ }v_k$. A node of a protocol Π is *regular* if it lies on a strand of Π, not on an adversary strand.

Definition 3 (Deliver-once)**.** *Suppose that S is a set of messages, and $\mathscr{B}$ is a bundle. $\mathscr{B}$* delivers messages in S only once *if there exists an injective function $f\colon R \to T$, where*

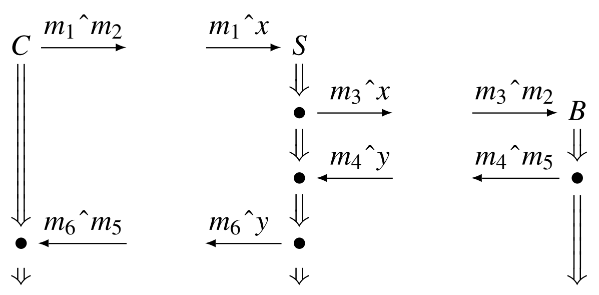
- *R is the set of regular nodes n in $\mathscr{B}$ such that a member of S is received on n, and*
- *T is the set of regular nodes n in $\mathscr{B}$ such that a member of S is transmitted on n.*

When $\{S_i\}_{i\in I}$ is a family of sets indexed by $i \in I$, we say that $\mathscr{B}$ is deliver-once *for $\{S_i\}_{i\in I}$ when $\mathscr{B}$ delivers messages in each S_i only once.*

We typically apply this definition when I is a set of values that will be generated freshly, and S_i is a set of messages of particular forms containing one such value i ($K_{j,k}$ in the example below).

Cryptoprotocol Example. The Buyer-Seller cryptoprotocol implements the choreography example of Section 3. It provides parametric strands that define the behaviors of the principals as they send and receive encrypted messages to provide security services for the behaviors in the choreography. The central idea is that the first few messages use public encryption keys and nonces to establish symmetric keys. The remaining messages then use the keys in a straightforward way. To establish a key between A and B, A sends a message containing a nonce, encrypted with B's public key. B returns a message encrypted with A's public key. It contains A's nonce as well as a fresh symmetric key to be used for this session. We use different syntactic tags in each encrypted unit which correspond to the op's in the choreography (denoted by the typewriter font `op`). At this level, the tags ensure that no unit can be confused with any other (this is the reason why the op's are all distinct at choreography level). The key exchange phase takes the form shown in Fig. 2. Each participant leaves the key exchange phase knowing that N_1, N_2 are shared among C, S, B, and that two symmetric keys are to be used for encryption in the next phase. For instance, C knows to use K_{sc} to communicate with the seller in the ensuing exchange, and to use K_{bc} to communicate with the bank.

In the ensuing stage, the participants use these keys to transfer the payloads amongst themselves. Their exchange—in the successful case, in which the transaction completes—takes the form shown in

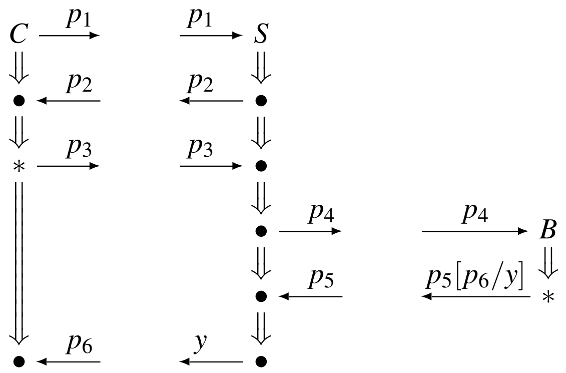


$m_1 = \{|\mathtt{cs}\ C\hat{}B\hat{}N_1|\}_{\mathsf{pubk}(S)}$ $m_2 = \{|\mathtt{cb}\ C\hat{}S\hat{}N_1|\}_{\mathsf{pubk}(B)}$
$m_3 = \{|\mathtt{sb}\ C\hat{}S\hat{}N_1|\}_{\mathsf{pubk}(B)}$ $m_4 = \{|\mathtt{bsk}\ N_2\hat{}N_1\hat{}K_{bs}|\}_{\mathsf{pubk}(B)}$
$m_5 = \{|\mathtt{bck}\ N_1\hat{}K_{bc}|\}_{\mathsf{pubk}(C)}$ $m_6 = \{|\mathtt{sck}\ N_1\hat{}N_2\hat{}K_{sc}|\}_{\mathsf{pubk}(C)}$

Figure 2: Key exchange phase

Fig. 3. However, C and B each have an opportunity to prevent the exchange from completing, at the nodes marked $*$. If C transmits $\{|\mathtt{refuse}|\}_{K_{sc}}$ instead of p_3, then S must terminate the exchange before contacting B. If B transmits $\{|\mathtt{nopaycf}\ \{|\mathtt{nopay}|\}_{K_{bc}}|\}_{K_{bs}}$ instead of $p_5[p_6/y]$, then S and C must terminate the transaction.

Analysis with CPSA indicates that this protocol achieves its goals. Let us assume that the participants of a run use their private decryption keys only in accordance with this protocol, and that the nonces N_1, N_2 and keys K_{bc}, K_{bs}, K_{sc} are in fact freshly chosen and unguessable. On this assumption, there are essentially only three possible executions, if we consider only those of minimal size, given that a role completed. When C completes normally, then the other participants have completed normally with matching parameters. When S completes with a client refusal, then C has refused and B has had a matching key exchange phase but no more. When C completes with a nopay message, then B has refused to pay, and S has been informed of this.



$p_1 = \{|\mathtt{req}\ N_2\hat{}C\hat{}S\hat{}B\hat{}\mathsf{prod}|\}_{K_{sc}}$ $p_2 = \{|\mathtt{reply}\ \mathsf{quote}|\}_{K_{sc}}$
$p_3 = \{|\mathtt{ok}\ x|\}_{K_{sc}}$ $p_4 = \{|\mathsf{pay}\ x\hat{}C\hat{}S|\}_{K_{bs}}$
$p_5 = \{|\mathtt{okcf}\ y|\}_{K_{bs}}$ $p_6 = \{|\mathtt{rcpt}\ \mathsf{receipt}|\}_{K_{bc}}$

Figure 3: Payload exchange phase

5 Abstraction and Correctness

A partial function α over messages is an *abstraction map* if (1) $\alpha(t)$ (if defined) contains no cryptographic operators, nonces nor keys, and (2) the parameters in $\alpha(t)$ (if defined) always appear in t.

For instance, α could map $\{|\mathtt{req}\, N_2\hat{\ }C\hat{\ }S\hat{\ }B\hat{\ }\mathsf{prod}|\}_{K_{sc}}$ to $\mathtt{req}\langle\mathsf{prod}\rangle$ in our Buyer-Seller example. The result has no cryptography and no nonces, and the tags req and prod appear in the argument.

We say that an abstract strand s is an *image* of a cryptographic strand s_c if, ignoring transmissions or receptions on s_c, for which α is undefined, for each transmission or reception node n on s, its message $\mathsf{msg}(n)$ is $\alpha(\mathsf{msg}(n_c))$, where n_c is the corresponding transmission or reception node (resp) on s_c. That is, α yielding the trace of s, when mapped through the trace of s_c restricted to the domain of α.

Suppose that a concrete strand s_c has its first i nodes in a concrete bundle $\mathscr{C}$, but α is undefined for the messages on these nodes. We then say that s_c is *abstractly vacuous in* $\mathscr{C}$. In the opposite case, when some node n of s_c is in $\mathscr{C}$ and $\alpha(\mathsf{msg}(n))$ is well-defined, we say that s_c is *abstractly non-vacuous in* $\mathscr{C}$.

An abstract bundle $\mathscr{B}$ is an *image* of a cryptographic bundle $\mathscr{C}$ if (1) there is a bijection ϕ between the abstractly non-vacuous regular strands s_c of $\mathscr{C}$ and the regular strands s of $\mathscr{B}$; (2) $\phi(s_c)$ is always an image of s_c; and (3) the transmission relation $\rightarrow_{\mathscr{B}}$ is formed by connecting nodes of $\mathscr{B}$ such that $m \rightarrow_{\mathscr{B}} n$ implies $m_c \preceq_{\mathscr{C}} n_c$, for some concrete nodes of which m, n are images. See [9] for a related notion of protocol transformation, and [1] for an approach to protocol verification via abstraction functions.

Suppose that $\mathscr{C}$ is a concrete bundle and $\{\mathscr{C}_i\}_i$ is a family of sub-graphs of $\mathscr{C}$ that partitions the regular nodes of $\mathscr{C}$. We say that $\{\mathscr{C}_i\}_i$ *separates* $\mathscr{C}$ *into components* when each $\mathscr{C}_i$ is a bundle on its own.

Definition 4 (Faithfulness)**.** *Cryptoprotocol* Π *is* faithful to *choreography C if there is an abstraction function* α *such that:*

1. *Every* $\mathscr{B} \in [\![C]\!]$ *is an image of some bundle* $\mathscr{C}$ *of* Π*;*

2. *If* $\mathscr{C}$ *is a bundle of* Π*, then some family* $\{\mathscr{C}_i\}_i$ *separates* $\mathscr{C}$ *into components. Moreover, each image* $\mathscr{B}_i$ *of any* $\mathscr{C}_i$ *is an initial sub-bundle of* $\sigma(\mathscr{B})$*, for some* $\mathscr{B} \in [\![C]\!]$ *and some substitution* σ*.*

If $\{S_i\}_{i\in I}$ *is a family of sets of messages, then* Π *is* faithful to C assuming the deliver-once property for $\{S_i\}_{i\in I}$ *if the above holds for bundles of* Π *that are deliver-once for* $\{S_i\}_{i\in I}$*.*

Faithfulness in the Buyer-Seller protocol. We use the protocol analysis tool CPSA [7] as part of a proof that the protocol of Fig. 2 and Fig. 3 is faithful to the choreography in Fig. 1. There are three stages:

1. CPSA determines the minimal, essentially different executions that are possible, given that any one party has had a complete run.

   These are the expected success execution $\mathbb{A}_s$ and failure execution $\mathbb{A}_f, \mathbb{A}_{f'}$, modulo the fact that a party never knows whether its last message was successfully delivered, if its last action is a transmission. In particular, the active parties agree on all parameters to the session.

2. Based on this CPSA output, inspection shows that Def. 4, Clause 1 is satisfied: Any run $\mathscr{B} \in [\![C]\!]$ is the abstraction of some concrete bundle $\mathscr{C}$.

3. Because $\mathbb{A}_s, \mathbb{A}_f, \mathbb{A}_{f'}$ are the only minimal forms of execution, every larger execution $\mathscr{B}_c$ is a (possibly non-disjoint) union of executions of these forms. That is, there is a family of maps $\{H_i\}_i$, where each H_i maps either $\mathbb{A}_s$ or $\mathbb{A}_f$ to some subset of the regular nodes of $\mathscr{B}_c$. Moreover, each regular node $n \in \mathscr{B}_c$ is the image of some node in $\mathbb{A}_s, \mathbb{A}_f$, or $\mathbb{A}_{f'}$ under at least one of the H_i.

   However, each pair of strands agrees on a pair of freshly chosen values, where each of them has chosen one of the values. This forces the range of H_i and H_j either to coincide or be disjoint. Hence Clause 2 is satisfied when we define the family $\{\mathscr{C}_i\}_i$ by saying that two nodes belong to the same $\mathscr{C}_i$ if they are both in the range of any one H_i.

6 Concluding Remarks

We have introduced two execution models, one for choreography (assuming no compromised participants) and one for cryptoprotocols with deliver-once assumptions. The abstract bundle semantics gives a set of bundles representing all the possible runs of the protocol described by a choreography. We have sketched a form of argument for proving that a cryptoprotocol is faithful to the ABS of a choreography.

In [4], we studied an abstract semantics for the choreography language presented here where roles can belong to compromised principals. The ideas of abstraction have yet to be extended to the compromised case and to a choreography language with infinite states. The work by Bhargavan et al. in [3, 6] is closely related to ours: they provide a compiler for generating ML code that can then be type-checked for verifying its security property. Their notion of faithfulness is guaranteed for the well-typed code generated from the source choreography.

In future work, we aim at developing systematic techniques for proving that certain transformations preserve all of the goals of a protocol, while achieving additional goals [9].

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] Karthikeyan Bhargavan, Ricardo Corin, Pierre-Malo Deniélou, Cédric Fournet, Karthikeyan Bhargavan, and James J. Leifer. Cryptographic protocol synthesis and verification for multiparty sessions. In *22nd IEEE Computer Security Foundations Symposium CSF*. IEEE CS Press, 2009.

[4] Marco Carbone and Joshua Guttman. Choreographies with secure boxes and compromised principals. In *Pre-proceedings of ICE'09*, 2009.

[5] 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.

[6] 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.

[7] Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Searching for shapes in cryptographic protocols. In *Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, number 4424 in LNCS, pages 523–538. Springer, March 2007. Extended version at URL:`http://eprint.iacr.org/2006/435`.

[8] Daniel Dolev and Andrew Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.

[9] 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)*, number 5511 in LNCS, pages 107–123. Springer, 2009.

[10] Joshua D. Guttman, Jonathan C. Herzog, John D. Ramsdell, and Brian T. Sniffen. Programming cryptographic protocols. In Rocco De Nicola and Davide Sangiorgi, editors, *Trust in Global Computing*, number 3705 in LNCS, pages 116–145. Springer, 2005.

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

[12] Roger Needham and Michael Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), December 1978.

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