# Establishing and Preserving Protocol Security Goals *

Joshua D. Guttman

November 5, 2012

**Abstract**

We take a model-theoretic viewpoint on security goals and how to establish them. The models are (possibly fragmentary) executions. Security goals such as authentication and confidentiality are implications over the *geometric fragment* of predicate logic, i.e. implications $\Phi \longrightarrow \Psi$ where $\Phi$ and $\Psi$ are built from atomic formulas without negations, implications, or universal quantifiers.

Security goals are then essentially statements about homomorphisms where the source is a minimal (fragmentary) model of the antecedent $\Phi$. If every homomorphism to a model representing a non-fragmentary, complete execution factors through a model in which $\Psi$ is satisfied, then the goal is achieved. This idea suggests validating security goals via a process of *information enrichment*.

This idea also clarifies *protocol transformation*. A protocol transformation preserves security goals when it preserves the form of the information enrichment process. We formalize this idea using simulation relations between labeled transition systems.

## 1 Introduction

We focus here on the *security goals* of cryptographic protocols, namely the properties that they try to achieve to protect their participants. We will examine a variety of actual protocol goals, observing that they share a common *logical form*. This logical form justifies a particular type of *search* that is capable of establishing a security goal, or providing counterexamples. The nature of this search also suggests a way to prove that some *protocol transformations* preserve security goals.

*Supported by the National Science Foundation under grant CNS-0952287. This paper combines and expands the material of [34, 35, 36].

**Logical form of protocol security goals.** First, we will examine the confidentiality and authentication goals that protocols are designed to achieve. They form a fairly wide palette in practical terms, but (formalizing confidentiality as non-disclosure rather than as indistinguishability) they have a logical form in common. Namely, they are implications between positive-existential formulas. This means that they have the form

$$\forall \overline{x} \,.\, (\Phi \longrightarrow \Psi) \tag{1}$$

where the formulas $\Phi, \Psi$ may use $\wedge, \vee$, and $\exists$, but not $\neg$, $\longrightarrow$, or $\forall$. The observation that authentication and confidentiality goals take this form has a long prehistory. However, the remainder of the paper shows that this logical form marks a boundary between goals that can be established using particular kinds of protocol analysis, and those which—like indistinguishability results—require an essentially different method.

**Protocol analysis as a search.** This logical form suggests our second topic. We regard the implication of Eqn. 1 as suggesting a search. We consider structures, i.e. first order models, over a particular signature. Some of these models represent complete executions, containing enough information to explain every event included. For instance, for every event in which a message is received, some event is already contained in the structure in which that message was sent. Other structures are incomplete, since some events may be unexplained. We will call these structures *fragments*, and the self-contained ones *realized fragments*.

From the point of view of this search, Eqn. 1 is achieved if, starting out with fragments that satisfy $\Phi$, and exploring their homomorphic images, we always find that $\Psi$ must become satisfied before any realized fragment (complete execution) is encountered. This ensures that every complete execution that satisfies $\Phi$ will also satisfy $\Psi$. Thus, protocol verification may be implemented as a search through fragments, moving through their homomorphic images.

**Transformations and soundness.** This search is informative for understanding protocol refinement and transformation. Since protocols are rarely designed from scratch, designers are always working new messages into existing protocols, or piggybacking new message ingredients, or using existing protocols as subprotocols.

Some of these transformations, when applied to a protocol satisfying some security goals of interest, yield protocols that no longer satisfy these goals. They are unsound transformations relative to these goals. Other transformations are sound relative to the goals, since the resulting protocol will always satisfy the goals that the input protocol did. A transformation from $\Pi_1$ to $\Pi_2$ is *sound* for a set $S$ of goal formulas if $\Pi_2$ achieves all of the goals in $S$ that $\Pi_1$ achieves.

Viewing protocol analysis as a search gives a way to separate sound and unsound protocol transformations. Since the search process for each protocol forms a labeled transition system (LTS) on fragments, we can ask whether one of these LTSs simulates another. A translation from $\Pi_1$ to $\Pi_2$ is sound if the

LTS for $\Pi_1$ simulates the LTS for $\Pi_2$, and the latter can progress whenever the former can.

The main content of this paper is to provide an integrated account of security goals and their model theory that leads to this conclusion about sound protocol transformations. Indeed, the single-protocol result on evaluating goals via search turns out to follow immediately from this soundness theorem for transformations. It is simply the instance for the identity transformation from $\Pi_1$ to itself.

## 1.1 Some approaches to protocol analysis

Putting aside approaches motivated by ideas in computational cryptography, and considering only symbolic methods, there are several main approaches to protocol analysis. Perhaps the most direct is theorem proving, following Paulson [51]. One constructs a theory expressing assumptions about what can happen in a run of a protocol, and proves theorems about these runs. We followed that approach in our original strand space work [59, 39], although in a new model, and without mechanizing the theorem proving, as Paulson did in Isabelle. A second approach is to construct a theory of the adversary's abilities, interacting with the regular protocol participants. If this theory includes at least as much as the adversary can actually do, and there is no proof that the adversary can construct a counterexample to a goal, then the protocol in fact meets that goal. This method underlies ProVerif [9], which constructs a set of Horn clauses over-approximating what the adversary can do; if resolution using these and the goal does not find a contradiction, then the goal is satisfied. This approach may also be represented fruitfully via type-checking [1, 28, 29]. Selinger developed the adversary-centered approach in a more model-theoretic form [57]; Goubault-Larrecq [30] subsequently used this idea to automatically extract proofs that can be mechanically checked.

Another approach works directly with representations of protocol executions, or partial executions. If protocols are represented within a process calculus, then their executions are traces for the process terms. Thus, a model-checker can examine these traces systematically, at least for small numbers of replications of the protocol roles, seeking misbehaviors. This approach dates back to Casper [43]. Alternatively, for bounded numbers of sessions, one can explore the possible executions using symbolic constraints, to observe whether misbehavior is possible [2, 27, 48, 56].

A variant of this approach applies to unbounded numbers of sessions. We will call it the *enrich-by-need* method. Given a partial execution, (i.e. a fragment, in our current vocabulary), one adds additional behaviors of protocol participants whenever this would help to complete the fragment to a full execution (i.e. a realized fragment). The enrich-by-need approach may not terminate, as the underlying class of problems in undecidable [25].

Enrich-by-need dates back to Millen [47] and Meadows [46], and is still at the center of Meadows's approach [26]. Song invented a form of enrich-by-need adapted to strand spaces in Athena [58]. It was subsequently redeveloped by

Cremers in Scyther [17]. Their approach introduces both regular behavior and adversary behavior as needed to explain message receptions that are already present in a fragment. Alternatively, CPSA [54], cf. [23, 37], uses the *authentication test* idea. The authentication tests are designed to factor out the adversary behaviors. Thus, CPSA works exclusively with skeletons, i.e. fragments containing no adversary actions.

The core goal of this paper is to give a logical treatment of enrich-by-need protocol analysis in strand spaces, focusing on how it represents and establishes security goal formulas, with an application to protocol transformation.

## 1.2 Our key ideas

We start by illustrating our key ideas with the simplest possible examples.

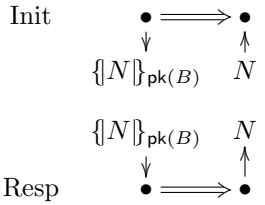



Figure 1: Protocol HD

**The Protocol HD.** HD, one of the simplest possible authentication protocols, is a half-duplex, authentication-only subprotocol of Needham-Schroeder [49]. It is shown in Fig. 1. HD gives the initiator an authentication guarantee that the responder has participated; it gives the responder no guarantee. HD does not establish any shared secret. The top half of the figure shows the initiator role, first transmitting the encrypted message for $B$ and then receiving the freed nonce $N$. The bottom half shows the responder role, first receiving the encrypted message and then freeing and transmitting $N$. A local run of HD is any instance of either one of these roles, using any values for the parameters $N, B$.

**Protocol analysis as a search.** To formalize authentication for HD, we consider a fragment of an execution in which there has been a local run of the initiator. In Fig. 2, we display schematically the result of plugging certain values into roles, which we treat as templates. Let us assume that $B$'s private decryption key $\mathsf{pk}(B)^{-1}$ is uncompromised, meaning that it will be used only in accordance with this protocol. We will also assume that in this run, $N$ was successfully chosen as fresh and unguessable. We write these assumptions $N \in \mathsf{unique}$ and $\mathsf{pk}(B)^{-1} \in \mathsf{non}$ for reasons we will describe in Section 2. The fragmentary execution we have just described is shown on the left of Fig. 2 as $\mathcal{F}$. It is, however, clearly incomplete. The value $N$ is sent inside an encryption when first created, and then is observed to have escaped from that container. The adversary cannot expect to come up with it independently; that is part of the assumption $N \in \mathsf{unique}$. The adversary cannot expect to extract it from $\{\!|N|\!\}_{\mathsf{pk}(B)}$; that is the assumption $\mathsf{pk}(B)^{-1} \in \mathsf{non}$.

Therefore a compliant participant must have received the message $\{\!|N|\!\}_{\mathsf{pk}(B)}$, and extracted $N$. In the protocol HD, this happens in only one way, namely in

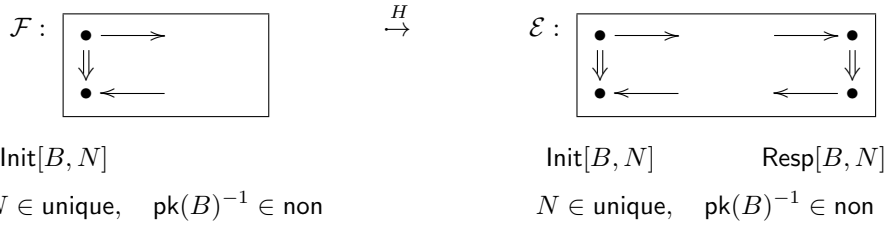



Figure 2: HD Authentication Guarantee

a local run of the responder. The result of adding this is shown on the right side of Fig. 2 as $\mathcal{E}$.

This chain of reasoning summarizes how CPSA analyzes HD [54]. It shows that any execution that has at least the structure shown on the left side has also at least the structure shown on the right side. This is an authentication result; the initiator's reception of $N$ authenticates that the responder $B$ has received the encryption with $N$ and extracted it. This process shows four suggestive characteristics:

1. The map $H$ is a structure preserving map $H\colon \mathcal{F} \to \mathcal{E}$, and we will introduce a notion of homomorphism that covers it (Defn. 2.6).

2. $H$ is more specifically a "problem-solution step." There is a problem in $\mathcal{F}$, namely to explain how $N$ escapes from the encryption $\{|N|\}_{\mathsf{pk}(B)}$. In this example, there is only one way to solve the problem, but in other examples that are several alternative solutions. We regard problem-solution steps as forming a labeled transition system. The nodes of the LTS are $\mathcal{F}, \mathcal{E}$, etc. The labels represent the problems. When there are several potential solutions, the LTS branches.

   Since there are different ways to organize the problem-solution process (e.g. CPSA uses a different idea from Athena and Scyther [58, 17]), we axiomatize the relevant properties of a problem-solution LTS in Defn. 6.3.

3. After solving this problem, there are no others to solve. We say that $\mathcal{E}$ is *realized*. However, many steps may be needed before reaching a realized diagram.

4. The authentication result that this process establishes is an implication. It says that if the structure on the left is observed, then all the structure on the right must in fact be present. This formula would say,

   **if** the initiator has taken both steps, in a local run with parameters $B$ and $N$, and pk$(B)^{-1} \in$ non, and $N \in$ unique,

   **then** there exist both steps of a responder local run using the same parameters.

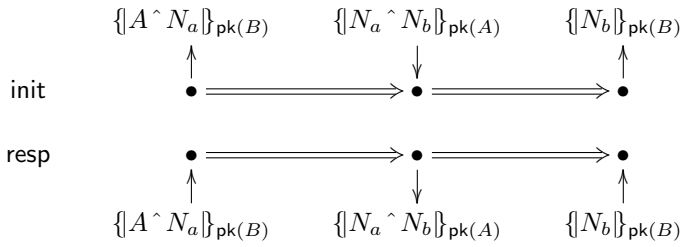



Figure 3: Needham-Schroeder Protocol NS

> This formula talks about the roles, and their parameters, and some properties of them, though it never says anything specific about the form of the messages. The forms of the messages are determined automatically by the protocol definition.

Cor. 6.5 formalizes the way that search in a problem-solution LTS can determine whether a protocol achieves a security goal of this kind.

**Transformations and soundness.** Our essential insight into protocol transformation is that an incremental design process—transforming simpler protocols into more complex ones—is sound when it preserves the structure of the source protocol's problem-solution LTS. Security goals will be preserved when the form of the protocol analysis process is preserved; the LTS is the embodiment of this protocol analysis process.

To illustrate that point, we now introduce a more complex protocol that reuses the ideas of HD.

**Needham-Schroeder and Needham-Schroeder-Lowe.** We can view the Needham-Schroeder [49] and Needham-Schroeder-Lowe [42] protocols as elaborations of the HD pattern. They consist of two roles, an *initiator* role and a *responder* role, which use public key cryptography to agree on a pair of fresh secret values. A session key for an encrypted conversation may be formed by combining them, for instance by hashing their concatenation. The messages exchanged in the protocol take the forms shown in Fig. 3. Lowe's corrected protocol NSL is identical, except that the responder's name $B$ appears in the second message, which therefore takes the form $\{|N_a \hat{} N_b \hat{} B|\}_{\mathsf{pk}(A)}$.

The parameters of each of these roles are $A, B, N_a, N_b$, of which the first two are principal names and the last two are nonces.

**Lifting an analysis.** We may superimpose HD on top of NS in two natural ways. First, we may associate the HD initiator with the first two steps of the NS initiator, and the HD responder with the first two steps of the responder.

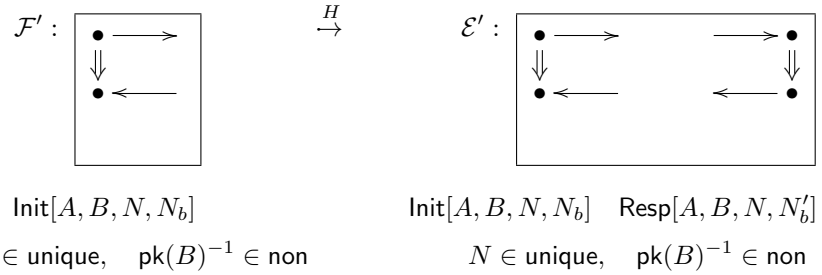



Figure 4: NS Initiator Authentication, lifting Fig. 2

In this mapping, transmissions are mapped to transmissions, and receptions are mapped to receptions. This mapping seems to explain the authentication that NS offers to the initiator.

Alternatively, we may associate the HD initiator with the second and third steps of the NS responder. The HD responder will map to the second and third steps of the NS initiator. This mapping also respects the direction of transmission and reception events.

We will explore the first of these two mappings now. The HD treatment of $N$ seems to explain how NS uses $N_a$ to achieve some authentication for the initiator. In the first case, it seems to explain how NS uses $N_b$ to achieve some authentication for the responder. Indeed, we can "lift" the fragment $\mathcal{F}$ from Fig. 2 to NS using either of our mappings. In the first mapping, we obtain the form shown in Fig. 4. On the left, we have simply copied the contents of Fig. 2, adjusting the HD initiator run to become the first two nodes of an incomplete NS initiator run. We have left the parameter instances $B, N$ unchanged. For the additional parameters of the NS initiator role, we have chosen unused, unconstrained values. The assumptions $N \in \mathsf{unique}, \mathsf{pk}(B)^{-1} \in \mathsf{non}$ are likewise carried over unchanged. On the right, we have done the same with both local runs. In particular, the responder strand has an unconstrained parameter, for which we have chosen the unused value $N'_b$, making no assumption that $N_b = N'_b$.

Curiously, Fig. 4 may also be interpreted directly as a problem-solution step in NS. Fragment $\mathcal{F}'$ has a problem, namely to explain how $N$ is received in the second node outside the encryption $\{|A \hat{} N|\}_{\mathsf{pk}(B)}$, having been sent in this form only. The adversary cannot achieve this on his own, since the decryption key is assumed uncompromised, $\mathsf{pk}(B)^{-1} \in \mathsf{non}$. The protocol provides only one way that a compliant principal with extract $N$ from an encryption of the form $\{|A \hat{} N|\}_{\mathsf{pk}(B)}$, namely the responder's transmission. The form shown in $\mathcal{E}'$ is thus the most general way to solve this problem.

This phenomenon, that the result of lifting a problem-solution step should yield a problem-solution step in the transformed protocol is the core observation

in our treatment of sound transformations. We formalize it in terms of the LTSs of the two protocols. If $\Pi_1$ is the source protocol and $\Pi_2$ is the target protocol, we want two properties to hold:

1. When a configuration in $\Pi_1$ has a problem and therefore needs a solution, the lifted configuration in $\Pi_2$ should have a corresponding problem;

2. Given corresponding problems, each solution in $\Pi_2$ should be the lifting of some solution in $\Pi_1$.

The first says that $\Pi_2$ should provide enough problems; the latter says that it should solve them only in corresponding ways. The first is a *progress* condition on the lifted LTS; the latter says that the original LTS should be able to *simulate* the behavior of the lifted LTS, at least with regard to corresponding problems. $\Pi_2$ may of course pose additional problems that will also have to be solved before possible executions are found. (See Thm. 7.8.)

There is one other major layer in this paper, as motivated by characteristic 4, p. 5. That is to provide a logical formalism to express the security goals of authentication, confidentiality, forward secrecy, etc. This language needs to be selected so that the goals can be expressed in forms that will respect protocol transformations. The example we have just considered tells us that the goal language should focus on transmission and reception events, and their parameters. It should remain mute on the forms of messages, since we would like to allow message forms to change as much as possible in protocol transformations. Indeed, what we have just learnt is that it is the problems and solutions that must be preserved, for soundness, not necessarily the forms of messages. The resulting languages $\mathcal{GL}(\Pi)$ match very well with the LTS of problems and solutions (as formalized in Cor. 6.5), and with the idea that sound protocol transformations should lift analyses (Thm. 7.12).

Reifying the protocol analysis activity into LTSs, and explaining relations between protocols using them, are new in this paper.

Although our results help us avoid verifying the transformed protocol directly, they also have a deeper value. They suggest design principles for incremental construction of protocols. We expect future work to lead to *syntactic* conditions that imply that a transformation preserves security goals. Protocol design, now a hit-or-miss activity requiring experience and ingenuity, could become a more predictable and possibly tool-supported process.

**Structure of this paper.** We start (Section 2) by introducing the terminology for the strand space model on which we will rely, illustrating it with the familiar Needham-Schroeder and Needham-Schroeder-Lowe protocols. In Section 3, we survey a wide variety of security goals, including various flavors of confidentiality and authentication, showing how they can be viewed as falling into the form of Eqn. 1, and illustrating the central role of homomorphisms in formulating security goals.

Section 4 introduces the first order languages of goals explicitly, and provides their semantics. Implicit in our examples are the notions of a *characteristic for-*

*mula* for a fragment, and a *characteristic fragment* for a formula. We introduce these notions in Section 5. With this in hand, we can justify enrich-by-need protocol analysis as a method for resolving security goals of the form given in Eqn. 1 (Section 6). We complete our program with a study of protocol transformations in Section 7. Some retrospective comments on what we have found, and its relation to other results, and found in Section 8.

## 2 Strands, Bundles, Fragments

We here summarize oft-used strand space vocabulary, and introduce the new notion of a *fragment*. As a familiar example, we use the Needham-Schroeder protocol [49]. For more information about strands and protocol analysis, see [37].

**An Algebra of Messages.** In this paper, we will regard the messages as forming an algebra ALG. Many alternatives to this particular message algebra are possible. ALG is an order-sorted algebra with the following six sorts. The first five are disjoint subsets of the last:

**principal names,** used to name protocol participants;

**nonces,** used when a principal chooses a value intended to be fresh;

**data,** used to represent payloads and other auxiliary message components;

**symmetric keys,** for ciphers;

**asymmetric keys,** used for asymmetric operations such as public key cryptography and digital signatures;

**messages,** a top sort which includes all values in ALG.

ALG populates each of these sorts with an infinite supply of primitive values. We will refer to the primitive values of sort *message*

We assume that asymmetric keys are equipped with an *inverse* operation, such that $(K^{-1})^{-1} = K$. Two asymmetric keys form a key pair if they are inverses of each other.

We extend the asymmetric keys by the two constructors, $\mathsf{pk}(A)$ and $\mathsf{privk}(A)$ which for any principal name $A$, returns a distinct asymmetric key disjoint from the parameters. They represent $A$'s *public encryption key* and *private signature key*, respectively. Their inverses are respectively $A$'s private decryption key and the public verification key for $A$'s signed messages. It is easy to augment these constructors with others, e.g. for the symmetric long term key shared between two participants in a shared-key protocol such as Kerberos. We write these key constructors in sans serif font.

These values—apart from the indeterminates—are the *basic values*. That is, a basic value is a name, nonce, or datum, or else a symmetric or asymmetric key. These keys include both parameters and also the range of a key constructor.

The algebra of messages ALG is built from the basic values and indeterminates by the following operations, which act freely:

**pairing,** the pair of $t_0$ and $t_1$ being written $t_0 \hat{} t_1$;

**encryption,** the encryption of $t_0$ using $t_1$ being written $\{|t_0|\}_{t_1}$;

**hashing,** the hash of $t$ being written $\mathsf{hash}(t)$; and

**digital signature,** the digital signature of $t_0$ using $t_1$ being written $[\![\, t_0 \,]\!]_{t_1}$.

We extend the key inverse operation to all messages by stipulating that if $t$ is any message other than an asymmetric key, then $t^{-1} = t$. The encryption $\{|t_1|\}_{t_2}$ is a symmetric encryption if $t_2$ is a symmetric key, meaning that ${t_2}^{-1} = t_2$ may also be used to decrypt a message of this form.

We will assume for now that some public key infrastructure allows each principal to determine the other's public encryption key.

For uniformity, we often regard a hash $\mathsf{hash}(t)$ as if it were the hash of a well-known value $v$ using $t$ as a key, i.e. $\{|v|\}_t$. We also often regard a digital signature $[\![\, t_0 \,]\!]_{t_1}$ as a pair $\{|\mathsf{hash}(t_0)|\}_{t_1} \hat{} t_0$ of an encrypted hash with the message $t_0$. To verify the signature means to decrypt the first component and check that the result matches the hash of the second component. With these representations in mind, we will often ignore digital signatures and hashes in proofs below, concentrating only on encryptions and pairs.

*Substitutions* $\alpha$ are functions on parameters. A substitution maps indeterminates to any messages in ALG; it maps nonces, data, and key parameters to values of the same sorts. The action of $\alpha$ on a message $t$, producing $\alpha(t)$, is defined by extending it homomorphically through the operators of $t$, subject to the rule that $(K^{-1})^{-1} = K$.

This algebra has the *most general unifier* (mgu) property [55]: If two messages $t_0, t_1$ have a common instance $\alpha(t_0) = \alpha(t_1)$, then there is a most general solution $\alpha_0$. This means that $\alpha_0(t_0) = \alpha_0(t_1)$, and for every common instance $\gamma(t_0) = \gamma(t_1)$, $\gamma = \beta \circ \alpha_0$ for some $\beta$.

**Strands via Needham-Schroeder.** For the sake of simplicity, we will use the familiar Needham-Schroeder [49] and Needham-Schroeder-Lowe [42] protocols. (See Fig. 3.)

The parameters of each of these roles are the basic values $A, B, N_a, N_b$, of which the first two are principal names and the last two are nonces.

Each row of bullets, connected by double arrows $\bullet \Rightarrow \bullet \cdots$ is a *strand*, representing the sequence of message transmissions and receptions in a single local session of the protocol. The direction of the associated single arrow $\rightarrow$ distinguishes transmissions from receptions. The message patterns (written here on the top and bottom lines) indicate contents to be sent or received. We write $\mathsf{msg}(n)$ for the message sent or received on the node $n$, while $\mathsf{dmsg}(n)$ is its direction and message, which we write $+t$ for transmission of $t$ and $-t$ for reception of $t$.

We write $s \downarrow i$ to refer to the $i^{\text{th}}$ node of the strand $s$. We assume that messages, strands, and nodes all form pairwise disjoint sets.

We regard each of the two strands separately as a template. The *instances* of the template are the sequences of messages we obtain if we fill in suitable values in place of its parameters $A, B, N_a$, and $N_b$. The instances are called strands also; a strand serving as template is called a *role*.

We assume that the parameters such as $A, B, N_a, N_b$ are written in a fixed order for any particular role. The instances of a role are all the strands that result by applying substitutions $\alpha$ that specify, for each parameter $v$, what value $\alpha(v)$ to replace it with.

We often label a strand using notation like $\mathsf{Init}[A, B, N_a, N_b]$, indicating what role it is an instance of, and the parameters, or what values have been substituted for the parameters. A pair of strands $\mathsf{Init}[A, B, N_a, N_b]$ and $\mathsf{Resp}[C, D, N_a, N'_b]$ have the same value selected for the third parameter, but different values for the remaining parameters.

We also assume that every protocol contains one special role that we will call the *listener* role. The listener role consists of a single reception node $\bullet \stackrel{x}{\Leftarrow}$ which can receive any message as an instance of the indeterminate parameter $x$. The purpose of a listener strand is to witness for the fact that the message $x$ was available to be received. For instance, if the message $y$ was intended to be a secret, then the listener strand $\bullet \stackrel{y}{\Leftarrow}$, which instantiates the listener role with this message $y$, witnesses for the fact that message $y$ was available as transmitted without any cryptographic protection. Thus, they are useful for expressing confidentiality goals.

A particular execution may include any number of instances of one of these roles, or of their initial segments, e.g. the first two nodes of the initiator or responder role. And it may contain the same or a different number of instances—with the same or different values plugged in—of the other role. These non-adversary strands, in which a compliant principal follows the protocol, are called *regular strands*.

**Definition 2.1** $\Pi$ *is a* protocol *iff* $\Pi$ *is a finite set of strands* $\rho$*, called the* roles *of the protocol, including the listener role. We assume that if* $x$ *is an indeterminate (parameter of message type) and* $x$ *occurs in a transmission node* $\rho \downarrow i$*, then for some reception node* $\rho \downarrow j$ *with* $j < i$*,* $x \sqsubseteq \mathsf{msg}(\rho \downarrow j)$*. That is, parameters of unconstrained message type are acquired on reception nodes.*

$\Pi$ *also associates two sets of parameters of base sort with a role* $\rho$*, called* $\mathsf{role\_unique}(\rho)$ *and* $\mathsf{role\_non}(\rho)$*.*

We use $\mathsf{role\_unique}(\rho)$ and $\mathsf{role\_non}(\rho)$ in defining when a partial execution ("fragment") is permitted under the protocol $\Pi$ (Defn. 3.2).

Fix a protocol $\Pi$ such as Needham-Schroeder for the rest of this section. NS and all the protocols we discuss here use $\mathsf{role\_unique}(\rho) = \mathsf{role\_non}(\rho) = \emptyset$. If $\rho \in \Pi$ is a role, then $\mathsf{params}(\rho)$ is the set of parameters occurring in $\rho$.

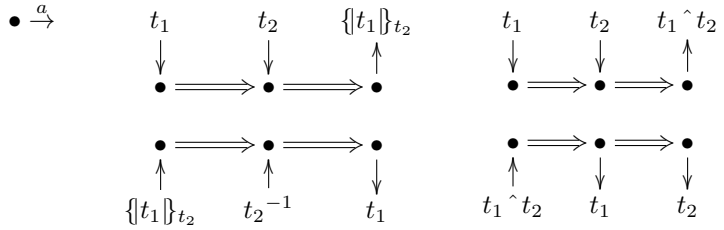



Figure 5: Adversary Roles: Generate basic value; encrypt and decrypt; concatenate and separate

**The Adversary.** An execution may also contain various instances of certain *adversary roles* that codify the basic abilities of the adversary.

The adversary (see Fig. 5) may originate a basic value $a$, in a one-node strand $\bullet \xrightarrow{a}$. It may also engage in a three-node strand that receives a value $t_1$ to be used as plaintext; then a value $t_2$ to be used as encryption key; and then transmits the encryption $\{|t_1|\}_{t_2}$. Another adversary role receives an encrypted value and its corresponding decryption key, after which it transmits the enclosed plaintext. The decryption key corresponding to $K$—written $K^{-1}$—is equal to $K$ if $K$ is a symmetric cryptographic key. If $K$ is one member of an asymmetric key pair, then $K^{-1}$ is the other key in this pair.

Adversary strands can also pair together two received messages; or, having received a pair, transmit each component of the pair separately. The adversary can also produce a digital signature given the signature key and the message to sign, and can verify a signature and retrieve the signed message. It can generate a hash for a given message.

**Bundles.** An execution is pieced together from a finite set of these regular and adversary strands (or their initial segments). Two nodes may be connected with a single arrow $\bullet \rightarrow \bullet$ when the former transmits a message, and the latter receives the same message directly from it. This leads to the notion of bundle, meaning a causally well founded graph built using strands:

**Definition 2.2** *Let $\mathcal{G} = \langle \mathcal{N}, \rightarrow_E \cup \Rightarrow_E \rangle$ be a finite, directed acyclic graph where (i) $n_1 \Rightarrow_E n_2$ implies $n_1 \Rightarrow n_2$, i.e. that $n_1, n_2$ are successive nodes on the same strand $n_1 = s \downarrow i$ and $n_2 = s \downarrow i + 1$; and (ii) $n_1 \rightarrow_E n_2$ implies that $n_1$ is a transmission node, $n_2$ is a reception node, and $\mathsf{msg}(n_1) = \mathsf{msg}(n_2)$.*

*$\mathcal{G}$ is a* bundle *if:*

1. *If $n_1 \Rightarrow n_2$, and $n_2 \in \mathcal{N}$, then $n_1 \in \mathcal{N}$ and $n_1 \Rightarrow_E n_2$; and*
2. *If $n_2$ is a reception node, there exists a unique $n_1 \in \mathcal{N}$ such that $n_1 \rightarrow_E n_2$.*

*$\mathcal{G}$ is an* open bundle *if:*

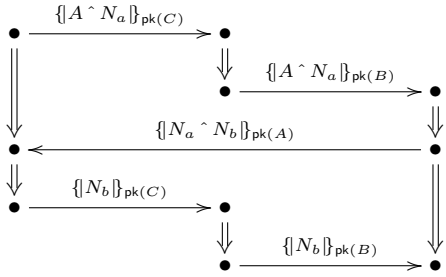



Figure 6: A Bundle in the NS Protocol, with adversary actions compressed

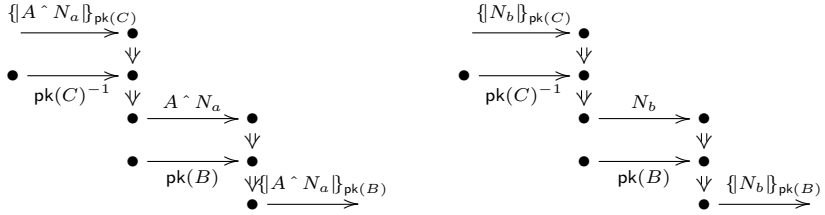



Figure 7: Adversary strands for Fig. 6

1. *If* $n_1 \Rightarrow n_2$, *and* $n_2 \in \mathcal{N}$, *then* $n_1 \in \mathcal{N}$ *and* $n_1 \Rightarrow_E n_2$; *and*
2. *If* $n_2$ *is a reception node, there is at most one* $n_1 \in \mathcal{N}$ *such that* $n_1 \rightarrow_E n_2$.

*If* $\mathcal{G}$ *is a bundle or open bundle, then* $\preceq_{\mathcal{G}}$ *is the causal partial order of reachability, defined to equal* $(\rightarrow_E \cup \Rightarrow_E)^*$, *and* $\prec_{\mathcal{G}}$ *is the strict partial order* $(\rightarrow_E \cup \Rightarrow_E)^+$.

The principle of bundle induction is crucial for all reasoning about protocols:

**Proposition 2.3** *Suppose that* $\mathcal{B} = \langle \mathcal{N}, \rightarrow_E \cup \Rightarrow_E \rangle$ *is a bundle, and* $S \subseteq \mathcal{N}$ *is a non-empty set of nodes. Then* $S$ *contains* $\preceq_{\mathcal{B}}$*-minimal elements.*

We often write $n \in \mathcal{G}$ when we mean $n \in \mathcal{N}$, where $\mathcal{G} = \langle \mathcal{N}, \rightarrow_E \cup \Rightarrow_E \rangle$. If every node $n$ on a strand $s$ is in $\mathcal{G}$, then we say that $s$ *has full height* in $\mathcal{G}$. The $\mathcal{G}$*-height* of $s$ is the number of nodes on $s$ that are in $\mathcal{G}$.

For an example bundle, see Fig. 6, in which however for want of space we have not written out the adversary strands in full. The necessary adversary strands are shown separately in Fig. 7. In each part of Fig. 7, four strands are shown. Two are of length 1, and represent the adversary transmitting to himself keys he knows, namely his own private decryption key $\mathsf{pk}(C)^{-1}$ and $B$'s public encryption key $\mathsf{pk}(B)$. The other two strands are first a decryption strand and then an encryption strand. This is typical of the way that the primitive adversary strands fit together to build up useful attacks.

**Ingredients and Origination.** We use a notion of message contents that covers the plaintext but not the key in an encryption. We write $\sqsubseteq$ ("is an ingredient of") for the smallest reflexive, transitive relation such that:

1. $t_1 \sqsubseteq (t_1 \hat{} t_2)$ and $t_2 \sqsubseteq (t_1 \hat{} t_2)$;
2. $t_1 \sqsubseteq \{\!|t_1|\!\}_{t_2}$.

By contrast, $t_2 \not\sqsubseteq \{\!|t_1|\!\}_{t_2}$ unless (anomalously) $t_2 \sqsubseteq t_1$.

We say that $t$ *originates* on a node $n$ if $n$ is a transmission node, and $t \sqsubseteq \mathsf{msg}(n)$, and for all $n_0$, if $n_0 \Rightarrow^+ n$, then $t \not\sqsubseteq \mathsf{msg}(n)$.

Thus, $t$ originates at $n$ if $n$ transmits $t$, and $t$ was neither transmitted nor received earlier along the strand of $n$. We regard a basic value $a$ as occurring *freshly* in a bundle $\mathcal{B}$ if it originates at just one node of $\mathcal{B}$. In this case, $a$ was chosen by a participant, without having the bad luck that any other principal selected the same value independently. We call a basic value $a$ *uniquely originating* in $\mathcal{B}$ if there exists exactly one node $n \in \mathcal{B}$ such that $a$ originates at $n$.

A key is regarded as uncompromised in $\mathcal{B}$ if it originates *nowhere*. We call a basic value $a$ *non-originating* in $\mathcal{B}$ if there exists no node $n \in \mathcal{B}$ such that $a$ originates at $n$. It may still be used in $\mathcal{B}$ even if it does not originate anywhere, since the regular strands may receive and send messages encrypted by $K$ or $K^{-1}$, thus using $K$ for encryption and decryption (resp.). Identifying non-originating keys with uncompromised ones is justified by this proposition:

**Proposition 2.4** *Suppose that $\mathcal{B}$ is a bundle, and $n \in \mathcal{B}$. If $t \sqsubseteq \mathsf{msg}(n)$, then there exists an $m \in \mathcal{B}$ such that $t$ originates on $m$. Cf. [59].*

*Suppose $K$ is non-originating in bundle $\mathcal{B}$. There is no encryption strand of full height in $\mathcal{B}$ which transmits any message of the form $\{\!|t|\!\}_K$. There is no decryption strand of full height in $\mathcal{B}$ which receives any message of the form $\{\!|t|\!\}_{K^{-1}}$ and transmits $t$.*

We use the terms *non-originating* and *uniquely originating* only in the case of basic values $a$, not compound values $t_1 \hat{} t_2$ or $\{\!|t|\!\}_K$.

We occasionally write $\ll$ for the smallest reflexive, transitive relation that is closed under the rules for $\sqsubseteq$ and also:

3. $t \ll \{\!|t_0|\!\}_t$.

Both $\ll$ and $\sqsubseteq$ are subrelations of the usual relation of *occurring in*, defined via the inductive generation of the messages. For instance, $A$ occurs in $\mathsf{privk}(A)^{-1}$, but $A \not\ll \mathsf{privk}(A)^{-1}$. In fact, if $a, b$ are basic values, then $a \ll b$ implies $a = b$.

**Fragments.** In protocol analysis, our aim is to find out what the protocol allows to happen in its bundles. However, to carry out protocol analysis conveniently, we would like to work with objects that are incomplete executions. We can then fill them in gradually to infer representatives of the different kinds

of protocol executions (bundles). We will call these objects *fragments*. Previous authors have called similar objects *semibundles* [58], *open bundles* [16], and *patterns* [17]. Our *skeletons*, of which much more later, are also related [37].

Although a fragment need not contain enough transmission events to be causally well-founded, it does impose constraints on what events may be added to obtain a relevant completing bundle. These are ordering constraints, which may impose an ordering on nodes that are not yet connected by a path; and origination constraints, i.e. that some basic values must remain at most uniquely originating, and that some must remain non-originating.

**Definition 2.5** *Let $\mathcal{G} = \langle \mathcal{N}, \rightarrow_E \cup \Rightarrow_E \rangle$ be an open bundle. Let $\preceq \subseteq \mathcal{N} \times \mathcal{N}$ be a partial order on the nodes. Let* $\mathsf{unique}, \mathsf{non}$ *be finite sets of basic values.*
$\mathcal{F} = \langle \mathcal{G}, \preceq, \mathsf{unique}, \mathsf{non} \rangle$ *is a* fragment *iff:*

1. $(\rightarrow_E \cup \Rightarrow_E) \subseteq \preceq$;
2. *If* $a \in \mathsf{unique}$ *then:*
   (a) *For some node* $n \in \mathcal{N}$, $a \sqsubseteq \mathsf{msg}(n)$;
   (b) *If* $a$ *originates at node* $n_0 \in \mathcal{N}$, *then*
      i. *if* $a$ *also originates at node* $n_1 \in \mathcal{N}$, *then* $n_0 = n_1$;
      ii. *if* $a \sqsubseteq \mathsf{msg}(n_1)$ *for* $n_1 \in \mathcal{N}$, *then* $n_0 \preceq n_1$;
3. *If* $a \in \mathsf{non}$, *then:*
   (a) *For some node* $n \in \mathcal{N}$, $a \ll \mathsf{msg}(n)$ *or* $a^{-1} \ll \mathsf{msg}(n)$;
   (b) *For all nodes* $n \in \mathcal{N}$, $a \not\sqsubseteq \mathsf{msg}(n)$;

*A fragment $\mathcal{F} = \langle \mathcal{G}, \preceq, \mathsf{unique}, \mathsf{non} \rangle$ is* realized *if $\mathcal{G}$ is a bundle.*

Since $\Rightarrow_E$ is completely determined by $\mathcal{N}$, we will often write a fragment $\langle \langle \mathcal{N}, \rightarrow_E \cup \Rightarrow_E \rangle, \preceq, \mathsf{unique}, \mathsf{non} \rangle$ in the form $\langle \mathcal{N}, \rightarrow_E, \preceq, \mathsf{unique}, \mathsf{non} \rangle$. We will also write $\mathsf{nodes}(\mathcal{F})$ to refer to the set $\mathcal{N}$ of nodes, and $\mathsf{regnodes}(\mathcal{F})$ to refer to the nodes that are regular, not adversary, nodes.

If $\mathcal{F}$ is a fragment, then the set $\mathsf{params}(\mathcal{F})$ of parameters of $\mathcal{F}$ contains all the images of parameters of roles $\rho \in \Pi$ that $\mathcal{F}$ uses. That is:

$$\mathsf{params}(\mathcal{F}) = \{\alpha(a) \colon (\alpha(\rho) \downarrow i) \in \mathsf{nodes}(\mathcal{F}) \wedge a \text{ occurs in } \rho \downarrow i\}.$$

**Homomorphisms between fragments.** A homomorphism is a structure-preserving map. In our case, we represent a homomorphism between fragments by two components, namely a substitution to be applied to the messages and a map from the nodes of the source to nodes of the target. The substitution determines the message on a target node, given the message on a source node.

**Definition 2.6** *Let $\mathcal{F}_1, \mathcal{F}_2$ be fragments, with $\mathcal{F}_1 = \langle \mathcal{N}_1, \rightarrow_1, \preceq_1, \mathsf{unique}_1, \mathsf{non}_1 \rangle$ and $\mathcal{F}_2 = \langle \mathcal{N}_2, \rightarrow_2, \preceq_2, \mathsf{unique}_2, \mathsf{non}_2 \rangle$. Let $\alpha$ be a substitution; and let $f$ be a map $f \colon \mathsf{nodes}(\mathcal{F}_1) \rightarrow \mathsf{nodes}(\mathcal{F}_2)$. Then $H = (f, \alpha)$ is a* homomorphism *if:*

1. *$n$ is a transmission node or respectively a reception node iff $f(n)$ is;*
2. *$\alpha(\mathsf{msg}(n)) = \mathsf{msg}(f(n))$;*
3. *$n_1 \Rightarrow n_2$ implies $f(n_1) \Rightarrow f(n_2)$;*
4. *$m_1 \Rightarrow f(n_2)$ implies for some $n_1$, $m_1 = f(n_1)$ and $n_1 \Rightarrow n_2$;*
5. *$n_1 \rightarrow_1 n_2$ implies $f(n_1) \rightarrow_2 f(n_2)$;*
6. *$n_1 \preceq_1 n_2$ implies $f(n_1) \preceq_2 f(n_2)$;*
7. *$\alpha(\mathsf{unique}_1) \subseteq \mathsf{unique}_2$;*
8. *$\alpha(\mathsf{non}_1) \subseteq \mathsf{non}_2$; and*
9. *If $a \in \mathsf{unique}_1$ and $a$ originates at $n_1 \in \mathcal{N}_1$, then $\alpha(a)$ originates at $f(n_1)$.*

*We write $H \colon \mathcal{F}_1 \to \mathcal{F}_2$ when $H$ is a homomorphism from $\mathcal{F}_1$ to $\mathcal{F}_2$.*

*When $\alpha, \alpha'$ agree on all the parameters appearing in $\mathcal{F}_1$, then $[f, \alpha] = [f, \alpha']$; i.e., $[f, \alpha]$ is the equivalence class of pairs under this relation.*

*We sometimes use $H$ to refer to its components, writing (e.g.) $H(n)$ to mean $f(n)$ or $H \circ \beta$ to mean $\alpha \circ \beta$, when $H = [f, \alpha]$ and $n \in \mathsf{nodes}(\mathcal{F})$.*

*$H = [f, \alpha]$ is an* inclusion map *if $f$ is the identity function. In this case, $\alpha$ is also the identity, i.e. $H = [\mathsf{Id}_{\mathsf{nodes}(\mathcal{F})}, \mathsf{Id}_{\mathsf{msgs}(\mathcal{F})}]$.*

*$\mathcal{F}$ is a* subfragment *of $\mathcal{E}$ if there is an inclusion $H \colon \mathcal{F} \hookrightarrow \mathcal{E}$.*

*$H$ is an* isomorphism *if there is a homomorphism $K$ such that $K \circ H = \mathsf{Id}$.*

*When $H = [f, \alpha] \colon \mathcal{F}_1 \to \mathcal{F}_2$, and $f$ is an injective function from $\mathsf{nodes}(\mathcal{F}_1)$ to $\mathsf{nodes}(\mathcal{F}_2)$, then we call $H$* node-injective*, and write $H \colon \mathcal{F}_1 \hookrightarrow_{ni} \mathcal{F}_2$*

The identity on a fragment $\mathcal{F}$ is always a homomorphism from it to itself, and the composition of two homomorphisms is a homomorphism.

Isomorphisms (or homomorphisms generally) depend only on the part of a strand that is actually in a fragment. Suppose, for instance, that a fragment $\mathcal{F}$ contains the first $i$ nodes of a strand $s$, and these nodes send and receive (respectively) the same messages that are sent and received on the first $i$ nodes of another strand $r$. The strands $s$ and $r$ may however do incompatible things after their first $i$ nodes. Suppose $\mathcal{F}'$ results from $\mathcal{F}$ when we surgically excise the nodes of $s$, and implant the corresponding nodes of $r$ in their place. Then this operation is an isomorphism from $\mathcal{F}$ to $\mathcal{F}'$, even if the $i+1$st node of $s$ is incompatible with the $i+1$st node of $r$.

In this sense, a fragment is only sensitive to the transmit/receive behavior of the nodes that lie within it. The fragment is not sensitive to what role these nodes were instantiated from. In the example above, $s$ and $r$ may be instantiated from different roles, which behave differently after the first $i$ nodes, but this role may differ in isomorphic fragments.

# 3 Security Goals: Some Examples

In this section, we will examine some examples of security properties, showing that they may be viewed as implications of the form of Eqn. 1. These examples will also clarify the underlying signature of the language of security goals.

We will set aside distinguishability properties. A distinguishability property is about the relation between different executions. To distinguish a real value $r$ from a randomly chosen fake value $f$ (for instance) the adversary considers whether—among executions compatible with his observations—those that use $r$ are more probable than those using $f$. This is not a property that is true or false in any one execution; rather, it is a property about the set of executions (with a distribution on that set). Thus, we cannot give a semantic treatment of distinguishability properties by considering individual executions of a protocol as models.

Indeed, our study in this paper is essentially defined as what we can establish about protocols by the formulas that are true in each individual execution, viewed as a model in the sense of first order logic.

This includes authentication; confidentiality in the sense of non-disclosure of values; and related properties of a single execution. Indeed, many relevant security goals can be expressed easily in statements of the form of Eqn. 1. They include forward secrecy, and resistance to attacks in which the adversary gets a regular participant to re-adopt an old, now-compromised key. They also include resistance to impersonation attacks, in which the adversary impersonates $B$ to $A$ when $B$'s long term secrets are not compromised, although $A$'s own secrets are. Implicit authentication, an important goal of some Diffie-Hellman protocols, is also of this form [24].

In this section, we will illustrate several of these properties, thereby explaining the vocabulary we work with, and how executions satisfy or provide counterexamples to formulas using that vocabulary. We use the Needham-Schroeder and Needham-Schroeder-Lowe protocols as our example protocols. We will also translate our diagrams into formulas, thereby introducing the protocol goal languages $\mathcal{GL}(\mathsf{NS})$ and $\mathcal{GL}(\mathsf{NSL})$.

**Authentication.** We can specify a classic authentication goal by giving two fragments, with a homomorphism between them. We start with a fragment representing an assumed configuration, e.g. that the responder has had a run of the protocol. In this configuration, we must also stipulate freshness and non-compromise assumptions. The assumptions that matter in this particular example are $N_b \in \mathsf{unique}$ and $\mathsf{pk}(A)^{-1} \in \mathsf{non}$. $B$'s authentication guarantee that $A$ has engaged in this session with him depends on $B$'s assumption that $A$'s private decryption key is uncompromised, and also that $B$'s nonce is fresh and unguessable. We illustrate the authentication goal if Fig. 8. There is a similar goal starting from an initiator strand.

Our convention is to box the strands in fragments, to set them off visually. Fig. 8 expresses the claim that—starting from the fragment on the left, containing just a responder strand and assuming freshness for $N_b$ and non-compromise

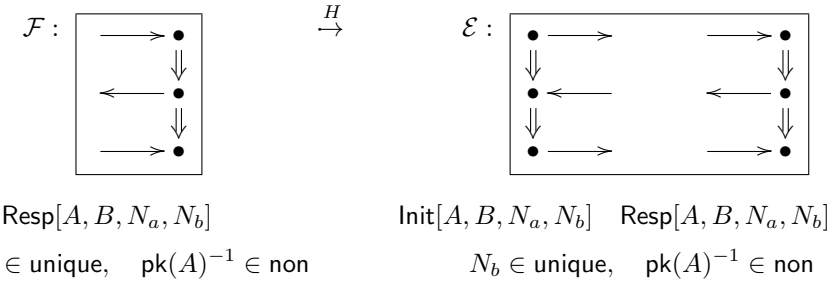



Figure 8: Responder's Authentication Guarantee

for $\mathsf{pk}(A)^{-1}$—every execution exhibits at least the structure shown in the fragment on the right. More formally, every homomorphism $J\colon \mathcal{F} \to \mathcal{D}$, where $\mathcal{D}$ is any *realized* fragment, factors through $H$. In particular, there exists some $K\colon \mathcal{E} \to \mathcal{D}$ such that $J = K \circ H$.

So any execution $\mathcal{D}$ which contains a responder strand with a fresh nonce and uncompromised peer also contains an initiator strand; moreover, the initiator strand and responder strand agree on the respective values for their parameters.

The *white space* in the diagram of $\mathcal{E}$ is, so to speak, pregnant, not empty. The goal is making an assertion about the possibly much larger execution $\mathcal{D}$, in which there may be many runs of the protocol. Some of these may be between $A$ and $B$; others, between one of them and a new principal $C$, possibly compromised; and yet others between new principals. Keys in these other executions may be compromised or not, and nonces may be sometimes freshly chosen and sometimes stale. Some sessions may be incomplete and some local runs unmatched. However, what Fig. 8 asserts is that these other sessions make no difference: No matter what goes wrong elsewhere, on the assumptions shown in $\mathcal{F}$, the desired behavior shown in $\mathcal{E}$ is sure—at least—to be present.

This security goal is not achieved by NS; indeed, Fig. 6 shows a counterexample. The map that embeds $\mathcal{F}$ into Fig. 6 cannot factor through $H$, because the initiator strand in Fig. 6 contains the parameter $C$ for its peer, in contrast to the responder's parameter $B$. In every homomorphic image of $\mathcal{E}$, the two strands agree on this parameter. Lowe's corrected protocol NSL [42] does achieve the goal, though.

We may express this authentication goal as a formula by writing out the contents of the two fragments in Fig. 8 in formulas. The fragment $\mathcal{F}$ on the left shows a contains a responder strand, with assumptions on two of its parameters. We will also mention its other parameters, because we will use them soon. Since a fragment containing the third node $n$ will necessarily contain $n$'s predecessors, we can simply say that the third node appears, thereby entailing that it also has two predecessors. However, we would like to mention the second node $m$, because we will assert that the nonce $N_b$ originates uniquely at $m$. Thus, this

formula characterizes $\mathcal{F}$:

$$\begin{gathered}\texttt{RespThd}(n) \wedge \texttt{RespScd}(m) \wedge \texttt{Coll}(m,n) \wedge \\ \texttt{Peer}(n,a) \wedge \texttt{Self}(n,b) \wedge \texttt{MyNonce}(n,c) \wedge \texttt{YourNonce}(n,d) \wedge \\ \texttt{Non}(\texttt{inv}(\texttt{pk}(a))) \wedge \texttt{UnqAt}(m,c).\end{gathered} \tag{2}$$

Here, $n, m$ are variables that in this formula refer to nodes; $a, b$ are variables that refer to principal names; and $c, d$ are variables that refer to nonce values. The first two atomic formulas say that $n, m$ are a third node of a responder strand and a second node of one. $\texttt{Coll}(m,n)$ says that they are "collinear" in the sense of lying on the same strand. The next line describes their parameters. The parameter $a$ represents the peer of $n$, and $b$ represents its own identity. The nonce $c$ is the one chosen on the strand itself, and $d$ is the one received on the first message. The last line expresses the assumptions on the key and nonce, namely that my nonce is uniquely originating and originates at node $m$, and that your private decryption key is non-originating.

All this is still true in $\mathcal{E}$ on the right, which also contains an initiator strand with matching parameters. We write this (incorporating Eqn. 2) in the form:

$$\begin{gathered}(2) \wedge \texttt{InitThd}(n') \wedge \\ \texttt{Self}(n',a) \wedge \texttt{Peer}(n',b) \wedge \texttt{YourNonce}(n',c) \wedge \texttt{MyNonce}(n',d)\end{gathered} \tag{3}$$

In particular, the security goal expressed in Fig. 8 is an implication, namely:

$$(2) \longrightarrow \exists n' \,.\, (3) \tag{4}$$

Observe that this formula is not valid in all fragments; for instance, its hypothesis is satisfied in $\mathcal{F}$, but its conclusion is not. However, can we make a *realized* fragment that satisfies Eqn. 2, but without also satisfying Eqn. 3?

If so, $\Pi$ achieves the goal in Fig. 8. NSL achieves it, though not NS.

NS does achieve something, namely the weaker authentication goal shown in Fig. 9. There is only one difference between Fig. 8 and Fig. 9, namely that the initiator's intended peer is any $B'$, not necessarily the same $B$. Fig. 9 yields the formula

$$(2) \longrightarrow \exists n', b' \,.\, (6) \tag{5}$$

where the conclusion uses the weakened formula:

$$\begin{gathered}(2) \wedge \texttt{InitThd}(n') \wedge \\ \texttt{Self}(n',a) \wedge \texttt{Peer}(n',b') \wedge \texttt{YourNonce}(n',c) \wedge \texttt{MyNonce}(n',d)\end{gathered} \tag{6}$$

in which the peer $b'$ may be different from $b$. This is too weak a goal if the protocol is also intended to preserve the secrecy of the nonces $N_a, N_b$, as we consider next.

This framework easily accommodates Lowe's insight that authentication properties are partially ordered. In fact, one goal is at least as strong as another if (in the first) the set of parameters shared between the original strand and the

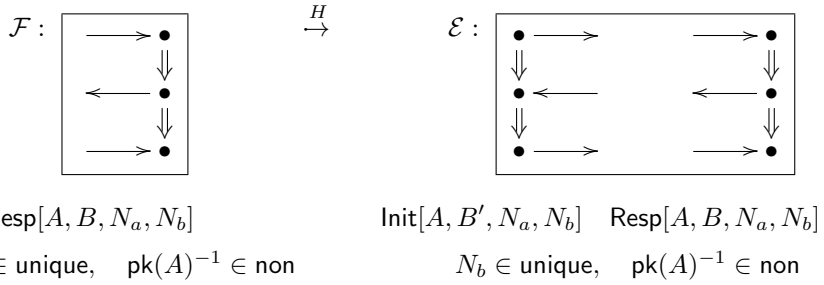



Figure 9: Responder's Weak Authentication Guarantee

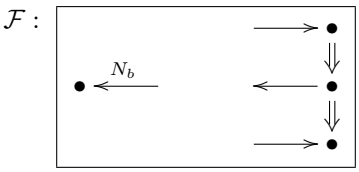
one whose presence is inferred is a subset of shared parameters in the second [44]. Thus, the partial order is simply the partial order of entailment among the right hand side (RHS) formulas, as they vary with a fixed assumption on the LHS. However, Lowe's injective agreement properties do not concern the same LHS, because they would then involve the quantifier "there exists uniquely," $\exists!$, in the conclusion, which is not positive existential. $\exists!$ contains a hidden universal quantifier. Instead, we find below that we can represent them with a different LHS.

In more complex protocols, disjunctive authentication goals are also possible. For a particular starting point $\mathcal{F}$, there may be a number of homomorphisms $H_1 \colon \mathcal{F} \to \mathcal{E}_1, \ldots, H_k \colon \mathcal{F} \to \mathcal{E}_k$ such that every execution compatible with $\mathcal{F}$ exhibits at least the structure shown in one of the fragments $\mathcal{E}_i$. More formally, every homomorphism $J \colon \mathcal{F} \to \mathcal{D}$, where $\mathcal{D}$ is any *realized* fragment, factors through one of the $H_i$; i.e. there exists some $i \leq k$ and some $K \colon \mathcal{E}_i \to \mathcal{D}$ such that $J = K \circ H_i$. Then a participant in the fragment $\mathcal{F}$ knows that—in any execution that his actions could be part of—all the facts in some $\mathcal{E}_i$ occur in this execution. This situations lead to authentication goals with disjunctive RHSs.

**Secrecy Goals.** To express secrecy goals, we make use of *listener* strands. We have assumed that every protocol contains a particular role, the *listener* role, which consists of a single reception node, receiving a message $x$, i.e. $\bullet \overset{x}{\leftarrow}$. To express that a value such as $N_b$ may be compromised in a given situation, we simply instantiate the listener role, and add a listener strand $\bullet \overset{N_b}{\leftarrow}$ to the fragment representing that situation.

Having formed a fragment $\mathcal{F}$ by adding a listener strand to express the compromise, we can assert that secrecy is preserved by saying that $\mathcal{F}$ does not extend to a realized fragment. That is, if $\mathcal{D}$ is any realized fragment, then $\mathcal{F} \not\to \mathcal{D}$.

As an example, Fig. 10 shows a fragment $\mathcal{F}$ containing a responder strand, with the assumptions that both private decryption keys are uncompromised, and that the nonce is freshly chosen and unguessable. By the symbol $\not\to \cdot$, we




$\not\rightarrow$ .

$\mathsf{Resp}[A, B, N_a, N_b]$

$N_b \in \mathsf{unique}, \quad \mathsf{pk}(A)^{-1}, \mathsf{pk}(B)^{-1} \in \mathsf{non}$

Figure 10: Secrecy for Responder's Nonce

mean to convey that there is no realized fragment $\mathcal{D}$ such that $\mathcal{F} \rightarrow \mathcal{D}$.

This security goal is in fact not achieved by NS, as Figs. 6–7 illustrate. That is, if we add a listener strand $\bullet \overset{N_b}{\leftarrow}$ to those figures, then we can clearly connect that strand to the middle transmission node on the right side of Fig. 7. This shows a way to map $\mathcal{F}$ into a realized fragment.

However, NSL does achieve the goal $\mathcal{F} \not\rightarrow \cdot$.

The essential difference between an authentication goal and a secrecy ("non-disclosure") goal is not the listener strand. Rather, it is the number 0. An authentication goal identifies a fragment $\mathcal{F}$ and 1 or more homomorphisms $H_1, \ldots, H_k$. It asserts that any homomorphism from $\mathcal{F}$ to a realized factors through one of the $H_i$. A non-disclosure goal is simply the case $k = 0$, in which homomorphism from $\mathcal{F}$ to a realized fragment factors through a member of the empty set of homomorphisms; i.e. there are none.

In fact, there are useful non-disclosure goals that do not involve listener strands. For instance, in a three-party protocol, two participants may agree on a value that should not be disclosed, neither to an outsider nor to the third party. The non-disclosure to an outsider is natural to express via a listener strand. The non-disclosure to the third party may be expressed by adding strands for this party with the sensitive value as one of the parameters. If these fragments do not extend to realized fragments, then the third party can not end up receiving the sensitive value in any execution.

Using the predicate $\mathtt{Lsn}(\ell)$ to say that $\ell$ is a listener node, and $\mathtt{Hear}(\ell, c)$ to say that the value heard on $\ell$ is $c$, we may express the content of the fragment $\mathcal{F}$ in Fig. 10 (again incorporating Eqn. 2) in the form:

$$(2) \wedge \mathtt{Lsn}(\ell) \wedge \mathtt{Hear}(\ell, c) \wedge \mathtt{Non}(\mathtt{inv}(\mathtt{pk}(b))) \tag{7}$$

and the secrecy goal has this as its premise, and asserts that it cannot occur. Its conclusion is the empty disjunction $\bot$.

$$(7) \longrightarrow \bot. \tag{8}$$

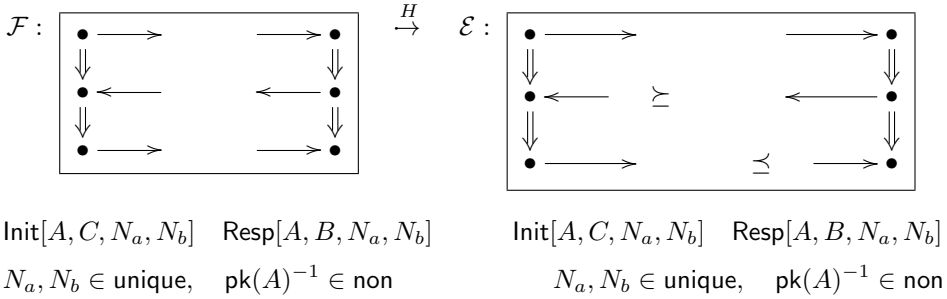



Figure 11: A Recency Goal for NS

It is achieved in NSL but not NS. That is, Eqn. 7 is satisfied in no *realized* fragment for NSL.

**Recent Key Generation.** Some protocols fail to ensure that a session key was recently generated. In these protocols, an adversary can cryptanalyze an old key, often a lengthy process, and then coerce a regular participant to re-adopt it for a new session [22]. Thus, one would often like to assert—and prove—that when a participant $B$ adopts its key, then not only does the intended peer $A$ have a matching local session, but that matching local session overlaps $B$'s session in time. The first part of this is an authentication goal, and the second part is a *recency* goal [31].

For this purpose, we use the fragment's causal partial order $\preceq$. We continue to choose our examples from the NS responder's point of view, in Fig. 11. Here, we assume a fragment $\mathcal{F}$ containing a responder strand and a partially matching initiator strand, as in the right side of Fig. 9. In this fragment, no ordering relations are assumed to hold between nodes on different strands. The conclusion asserts that the causal partial ordering holds between two pairs of nodes, ensuring that two nodes on the right hand strand "bracket" the corresponding nodes on the left hand strand. This recency goal is in fact achieved by the NS protocol.

Here, in the conclusion, what we are adding to the formula 3 are the two precedence assertions, and to express one of them, we need to be able to refer to the second node of the initiator strand.

$$(3) \wedge \texttt{InitScd}(m') \wedge \texttt{Coll}(m', n') \wedge \texttt{Preceq}(m, m') \wedge \texttt{Preceq}(n', n) \qquad (9)$$

The recency goal then takes the form:

$$(3) \longrightarrow \exists m' \,.\, (9) \qquad (10)$$

**Implicit Authentication.** Implicit authentication is an important goal of a group of protocols that use the Diffie-Hellman idea to achieve key agreement [8].

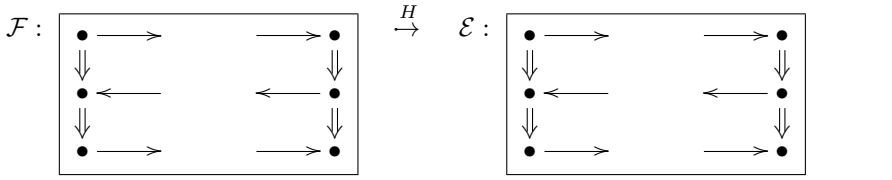



Figure 12: An Implicit Authentication Goal

The idea of implicit authentication is that *if* any peer shares the session secrets with me, *then* that principal is $A$. Thus, we again start with a fragment with two strands, as with our recency goal Fig. 11. However, in this case we will have the two strands agree on the session secrets, and have different name parameters for the principals. The security goal is for the protocol to ensure—in this case—that the strands must in fact agree on the principal names, as shown in Fig. 12. This asserts that every realized fragment must in fact identify $X = A$ and $Y = B$.

Here, we may express this by an implication in which the conclusion is a pair of equations. The hypothesis describes $\mathcal{F}$:

$$\begin{aligned} &\mathtt{RespThd}(n) \wedge \mathtt{RespScd}(m) \wedge \mathtt{Coll}(m, n) \wedge \mathtt{InitThd}(n') \wedge \\ &\mathtt{Peer}(n, a) \wedge \mathtt{Self}(n, b) \wedge \mathtt{MyNonce}(n, c) \wedge \mathtt{YourNonce}(n, d) \wedge \\ &\mathtt{Self}(n', x) \wedge \mathtt{Peer}(n', y) \wedge \mathtt{YourNonce}(n', c) \wedge \mathtt{MyNonce}(n', d) \wedge \\ &\mathtt{Non}(\mathtt{inv}(\mathtt{pk}(a))) \wedge \mathtt{UnqAt}(m, c) \wedge \mathtt{Unq}(d). \end{aligned} \tag{11}$$

It is enough to state $\mathtt{Unq}(d)$ because the initiator's nonce always originates on its first node, since this is a transmission node. Then the implicit authentication goal takes the form

$$(11) \longrightarrow x = a \wedge y = b \tag{12}$$

Although NS does not achieve this goal, NSL does achieve it. NSL also achieves the related goal where we assume $N_b \in \mathsf{unique}$ and $\mathsf{pk}(A)^{-1}, \mathsf{pk}(B)^{-1} \in \mathsf{non}$. That is, it suffices to assume both nonces uniquely originating, and the peer's decryption key is non-originating; or, alternatively, to assume that both decryption keys are non-originating, and one's own nonce is uniquely originating.

**Forward Secrecy.** A protocol offers forward secrecy if subsequent compromise of the principal's long term secrets cannot cause disclosure of an earlier session key. A protocol such as NS or NSL cannot achieve forward secrecy, because the private decryption keys are the only protection against disclosure of the secrets $N_a, N_b$. If, say, $\mathsf{pk}(A)^{-1}$ is subsequently disclosed, an adversary who has recorded the earlier encrypted messages will recover the secrets.

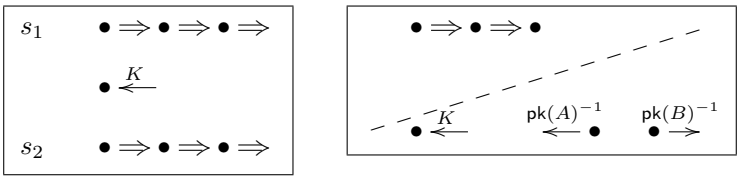



Figure 13: Weak and strong forward secrecy (resp.): these diagrams have no homomorphisms to realized fragments

Suppose we were to modify NSL to transport Diffie-Hellman values. The initiator $A$ would choose a value $x$, and use the exponentiated value $g^x \bmod p$ as its nonce $N_a$. The responder $B$ would choose a value $y$, and use the exponentiated value $g^y \bmod p$ as its nonce $N_b$. The protocol allows each principal to authenticate the source of its peer's value. If for a suitable prime $p$ and some reasonable $g < p$ they use

$$(N_a)^y \bmod p = K = (N_b)^x \bmod p$$

as a session key, $K$ will be shared only with the other. By the Decisional Diffie-Hellman assumption [10], subsequent compromise of the decryption keys discloses nothing about $K$.

There are two different ways that one might formalize the intuition, that subsequent compromise of the long-term keys should not disclose a session key $K$. Both of them are in fact secrecy goals, as they say that there are no homomorphisms from a fragment to any realized fragment.

One is that if two *regular* strands both yield the same session key $K$, then $K$ is not disclosed. On this formalization, the purpose of the long-term secrets is simply to ensure authentication, meaning that the two regular strands are present. So long as that holds, forward secrecy is the non-disclosure property saying that a listener strand for $K$ is impossible, as shown on the left in Fig. 13.

A somewhat stronger notion of forward secrecy stresses the word *subsequently*. One local session occurs, and the compromise of the long term keys happens after that session is finished: Can the adversary then retrieve the session key?

The easiest way to formalize this is to enrich the protocols with an additional role, the *blab* role $\bullet \xrightarrow{x}$ which simply transmits a value $x$. It is dual to the listener role.

Now, we formalize this idea in a diagram in which the long term secrets $\mathsf{pk}(A)^{-1}, \mathsf{pk}(B)^{-1}$ are transmitted after a session issuing in session key $K$ completes. Moreover, we assume that the long term secrets are *uniquely originating*. This implies that they cannot have been used before the session completed, which is exactly the intended force of considering a *subsequent compromise*.

Fig. 13 illustrates this situation on the right. The slanted dotted line separates past from future, meaning that any event northwest of the dotted line

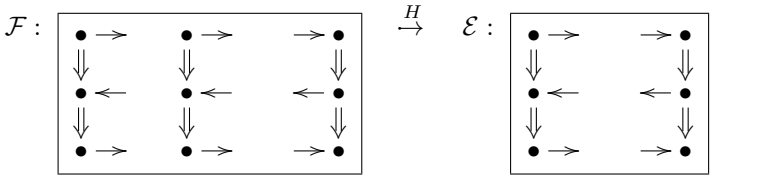



Figure 14: An Injectiveness Goal

occurs before any event southwest of it. This ordering relation between the end of the strand and the points of disclosure is essential to the idea.

We may express the content of this fragment in the formula:

$$\begin{gathered}(2) \wedge \mathtt{Lsn}(\ell) \wedge \mathtt{Hear}(\ell, k) \wedge \\ \mathtt{Unq}(\mathtt{inv}(\mathtt{pk}(a))) \wedge \mathtt{Unq}(\mathtt{inv}(\mathtt{pk}(b))) \wedge \\ \mathtt{Blab}(m_1) \wedge \mathtt{Said}(m_1, \mathtt{inv}(\mathtt{pk}(a))) \wedge \\ \mathtt{Blab}(m_2) \wedge \mathtt{Said}(m_2, \mathtt{inv}(\mathtt{pk}(b))) \wedge \\ \mathtt{Preceq}(n, m_1) \wedge \mathtt{Preceq}(n, m_2)\end{gathered} \tag{13}$$

where the goal is (13) $\longrightarrow \bot$.

**Injective agreement.** Lowe [44] also formalized the idea that there can be only one strand corresponding to a given strand, e.g. involving the same nonces. We formalize this as in Fig 14; it says that in any realized fragment that is an $H$-image of $\mathcal{F}$, the two strands on the left have been identified by the homomorphism $H$. Fig. 14 also asserts an equality, but, unlike implicit authentication, an equality between nodes:

$$\ldots \mathtt{InitThd}(n') \wedge \mathtt{InitThd}(n'') \ldots \quad \longrightarrow \quad n' = n''.$$

Although we have illustrated these types of goals with a specific pair of protocols, namely NS and NSL, these diagrams and their variants appear to capture a wide variety of properties that many protocols attempt to achieve.

Curiously, the diagrams in this section never involve adversary nodes. Instead, they use only regular nodes, which include listener nodes and blab nodes. Moreover, although they use the precedence order $\preceq$, they never involve single arrow links between nodes $\bullet \rightarrow \bullet$. Thus, they are *skeletons* [37]:

**Definition 3.1** *A fragment $\mathcal{F} = \langle \mathcal{N}, \rightarrow_E, \preceq, \mathsf{unique}, \mathsf{non} \rangle$ is a* skeleton *if (i) $\rightarrow_E = \emptyset$, and (ii) $\mathcal{N}$ contains only regular nodes (and no adversary nodes).*

*If $\mathcal{E} = \langle \mathcal{N}, \rightarrow_E, \preceq, \mathsf{unique}, \mathsf{non} \rangle$ is any fragment, then the skeleton $\mathsf{skeleton}(\mathcal{E}) = \langle \mathcal{N}', \emptyset, \preceq, \mathsf{unique}, \mathsf{non} \rangle$ where $\mathcal{N}' \subseteq \mathcal{N}$ contains only its regular nodes.*

*A skeleton $\mathbb{A}$ is a* realized skeleton *if there is some realized fragment $\mathcal{E}$ such that $\mathsf{skeleton}(\mathcal{E}) = \mathbb{A}$. We generally write skeletons in blackboard font, e.g. $\mathbb{A}, \mathbb{B}$.*

Thus, a skeleton is a realized skeleton if we can obtain a realized fragment by adding only adversary nodes to $\mathcal{N}$ and edges to $\rightarrow_E$.

Since all the security goals actually concern homomorphisms between skeletons, skeletons are thus the natural objects that security goals are talking *about*. Nevertheless, some approaches to protocol analysis build realized skeletons from given skeletons as "starting points" by generating fragments that are not skeletons along the way, including both regular and adversary activity.

A protocol *enforces* a security goal, represented via homomorphisms as in the figures above, if all *realized* fragments permitted by that protocol resulting from the starting point do so by a homomorphism that factors through the given homomorphisms.

**Definition 3.2** *Let* $\Pi$ *be a protocol.* $\mathcal{F}$ *is a* $\Pi$-fragment *if* $\mathcal{F}$ *is a fragment, and*

1. *if a regular strand* $s$ *has nodes in* $\mathsf{nodes}(\mathcal{F})$*, then* $s = \alpha(\rho)$*, for some substitution* $\alpha$ *and some role* $\rho \in \Pi$*;*
2. *if* $a \in \mathsf{role\_unique}(\rho)$ *and* $n = \alpha(\rho) \downarrow i \in \mathsf{nodes}(\mathcal{F})$*, then if* $\alpha(a)$ *occurs in* $\mathsf{msg}(n)$*, then* $\alpha(a) \in \mathsf{unique}(\mathcal{F})$*; and*
3. *if* $a \in \mathsf{role\_non}(\rho)$ *and* $n = \alpha(\rho) \downarrow i \in \mathsf{nodes}(\mathcal{F})$*, then if* $\alpha(a)$ *occurs in* $\mathsf{msg}(n)$*, then* $\alpha(a) \in \mathsf{non}(\mathcal{F})$*.*

$\{H_i\}_{i\in I}$ *is a* family of homomorphisms based in $\mathcal{F}$ *iff, for each* $i \in I$*,* $H_i \colon \mathcal{F} \mapsto \mathcal{E}_i$ *is a homomorphism with the source fragment* $\mathcal{F}$*.*

*Let* $\{H_i\}_{i\in I}$ *be a family of homomorphisms based in* $\mathcal{F}$*.*

$\Pi$ enforces $\{H_i\}_{i\in I}$ *iff every homomorphism from* $\mathcal{F}$ *to a realized* $\Pi$*-fragment* $\mathcal{D}$ *factors through some* $H_i$*. That is, iff, for every realized* $\Pi$*-fragment* $\mathcal{D}$ *and homomorphism* $K \colon \mathcal{F} \mapsto \mathcal{D}$*, there exists an* $i \in I$ *and a homomorphism* $J \colon \mathcal{E}_i \mapsto \mathcal{D}$ *such that* $K = J \circ H_i$*.*

Thus, $\Pi$ enforces a family of homomorphisms if every homomorphism from its source to a realized fragment has to "factor through" one of the homomorphisms in that family, or "go by way of" one of them. All the examples considered in this section use either a singleton or the empty set as the index set $I$, but that is certainly not always the case. The fragments $\mathcal{E}_i$ that are the targets of the $H_i$ are not required to be realized; we require only that if $\mathcal{D}$ is realized, then the homomorphism to it factors through an $H_i$.

# 4 The Security Goal Language of a Protocol

Our next task is to define a language $\mathcal{GL}(\Pi)$ in classical first order logic with equality for each protocol $\Pi$, to express goals like those considered in Section 3. The diagrammatic goals are all expressed by implications between geometric formulas of $\mathcal{GL}(\Pi)$, i.e. by formulas of the form of Eqn. 1. This allows us to verify and falsify them by exploring the homomorphisms from a particular fragment $\mathcal{F}$, observing the forms of realized fragment that they lead to.

We design our language not to express too much. We would like some protocol transformations to be able to preserve the security goals of a source protocol. Since these transformations may change the forms of the messages, we will set up $\mathcal{GL}(\Pi)$ so that it does not describe the forms of the messages. Since a transformation may add message transmissions and receptions, we ensure that $\mathcal{GL}(\Pi)$ expresses the main facts about nodes without needing to state their positions on strands. Instead, $\mathcal{GL}(\Pi)$ talks about which events on which regular roles have occurred, and which values were the instances of the parameters.

In addition, it has predicates to express the precedence ordering, and the unique and non properties, and equality, but not much more.

Our language concerns only the *nodes* and their *parameters* and the relations among them. It does not specifically talk about strands; it says only that some nodes lie on the same strand as each other (they are "collinear").

Given any formula of the form of Eqn. 1, we can use $\alpha$-renaming, quantifier rules, and the rules for disjunction on the left and conjunction on the right to transform it to a set of formulas of the form:

$$\forall \overline{x} \,.\, (\phi \quad \longrightarrow \quad (\exists \overline{y_1} \,.\, \psi_1) \vee (\exists \overline{y_2} \,.\, \psi_2) \vee \ldots \vee (\exists \overline{y_j} \,.\, \psi_j)) \tag{14}$$

where: (i) $\phi$ and each $\psi_i$ is a conjunction of atomic formulas, and (ii) $\overline{x}$ and each $\overline{y_i}$ are disjoint lists of variables. Null and unary disjunctions ($j = 0$ or $j = 1$) are permitted, where the null disjunction $\bot$ is the constantly false formula. For this reason, we will focus in this section on formulas (14).

We formulate $\mathcal{GL}(\Pi)$ as a single-sorted logic, since that is notationally simpler. In an implementation, such as the one by Ramsdell for CPSA [53], one would prefer instead a multi-sorted (or preferably order-sorted) logic; however, that would not simplify anything considered here.

$\mathcal{GL}(\Pi)$ says nothing about the structure of $\Pi$'s messages, so it can express goals that are preserved when message structure is transformed. It classifies nodes by which action they are, on which role, and how they instantiate the role's parameters.

The vocabulary of the language $\mathcal{GL}(\Pi)$ has two parts. One part is independent of $\Pi$, and is present in the language for every $\Pi$. It includes functions that relate principals to their keys, and keys to their inverses. $\mathcal{GL}(\Pi)$ contains function symbols $\mathtt{pk}(a)$, $\mathtt{sk}(a)$, and $\mathtt{inv}(k)$, for $a$'s public encryption key; $a$'s private signature key; and the inverse of a key $k$, i.e. the other member of an asymmetric key pair. This part of the language can easily be extended, e.g. with a function symbol $\mathtt{lts}(a, b)$ that takes two principal names as arguments, returning their long-term shared symmetric key, since some protocols such as Kerberos use one. We write these functions, as all the (non-variable) vocabulary of the goal language, in typewriter font, as shown in Table 1.

The protocol-independent part also includes the predicate symbols shown in Table 1. $\mathtt{Preceq}(m, n)$ expresses that node $m$ precedes node $n$. $\mathtt{Coll}(m, n)$ expresses that nodes $m$ and $n$ lie on the same strand. $\mathtt{Unq}(v)$ expresses that the basic value $v$ originates uniquely. $\mathtt{UnqAt}(n, v)$ expresses that the basic value $v$ originates uniquely, and originates at the node $n$. $\mathtt{Non}(v)$ expresses the non-

| | | | |
|---|---|---|---|
| Functions: | $\mathtt{pk}(a)$ | $\mathtt{sk}(a)$ | $\mathtt{inv}(k)$ |
| | $\mathtt{lts}(a, b)$ | | |
| Relations: | $\mathtt{Preceq}(m, n)$ | $\mathtt{Coll}(m, n)$ | $=$ |
| | $\mathtt{Unq}(v)$ | $\mathtt{UnqAt}(n, v)$ | $\mathtt{Non}(v)$ |

Table 1: Protocol-independent vocabulary of the languages $\mathcal{GL}(\Pi)$

origination of the basic value $v$. As always, $=$ is equality. All protocol languages use this vocabulary to express structural properties of fragments.

The protocol-specific vocabulary consists of two kinds of predicates. *Role position* predicates $R(n)$ assert that node $n$ is a node lying at a particular position on a strand that is an instance of that regular role. For instance, $\mathtt{RespFirst}(n)$ could assert that $n$ is an instance of the first node of a responder role, which means that it receives a message of the form $\{|A \hat{} N_a|\}_{K_B}$ for some values of the parameters.

The second kind of predicate concerns the values of the parameters. A *parameter* predicate $P(n, v)$ asserts that node $n$ is formed by instantiating a particular parameter of its role with the value $v$. The same parameter predicate may be used for different roles, so long as—whenever a node may be viewed as lying on instances of two different roles—it satisfies the same parameter predicates no matter of which role it is viewed as an instance.

**Example 1: $\mathcal{GL}(\mathsf{NS})$.** $\mathcal{GL}(\mathsf{NS})$ includes the protocol-independent vocabulary. Its protocol-specific vocabulary includes the role position predicates:

$$\begin{array}{ccc} \mathtt{InitFst}(n) & \mathtt{InitScd}(n) & \mathtt{InitThd}(n) \\ \mathtt{RespFst}(n) & \mathtt{RespScd}(n) & \mathtt{RespThd}(n) \\ & \mathtt{Lsn}(n), & \end{array}$$

meaning that $n$ is the first, second, or third node on an initiator or responder strand (resp.), and that $n$ is the reception node on a listener strand, which we assume present in all protocols. Only the regular roles of the protocol, not the adversary roles, are expressed here. $\mathcal{GL}(\mathsf{NS})$ also includes the parameter predicates:

$$\begin{array}{cc} \mathtt{Self}(n, v) & \mathtt{Peer}(n, v) \\ \mathtt{MyNonce}(n, v) & \mathtt{YourNonce}(n, v) \\ \multicolumn{2}{c}{\mathtt{Hear}(n, v).} \end{array}$$

The predicates $\mathtt{Self}, \mathtt{Peer}$ express the name of the current principal, and of its intended communication partner. $\mathtt{MyNonce}, \mathtt{YourNonce}$ express the value of the nonce created on this strand, and of the one received purportedly from the peer. $\mathtt{Hear}$ relates a listener node to the message it hears.

In the fragment $\mathcal{F}$ shown on the left in Fig. 8, the upper node $n$ satisfies $\mathtt{RespFst}(n)$, the middle node satisfies $\mathtt{RespScd}(n)$, and the lower node satisfies $\mathtt{RespThd}(n)$. The upper node $n$ and the name $v = A$ satisfy $\mathtt{Peer}(n, v)$, while $n$ and $v = B$ satisfy $\mathtt{Self}(n, v)$. The upper node $n$ and the nonce $v = N_a$ satisfy

| $\tau_r$ | 1 | 2 | 3 |
|---|---|---|---|
| Init | `InitFst` | `InitScd` | `InitThd` |
| Resp | `RespFst` | `RespScd` | `RespThd` |

| $\tau_p$ | $A$ | $B$ | $N_a$ | $N_b$ |
|---|---|---|---|---|
| Init | `Self` | `Peer` | `MyNonce` | `YourNonce` |
| Resp | `Peer` | `Self` | `YourNonce` | `MyNonce` |

Table 2: $\tau_r$ and $\tau_p$ for $\mathcal{GL}(\mathsf{NS})$

the parameter predicate $\texttt{YourNonce}(n, v)$. There is no $v$ such that this $n$ and $v$ satisfy $\texttt{MyNonce}(n, v)$, since my nonce has not yet been chosen at the time of $n$. However, the middle node $m$ and the nonce $v = N_b$ do satisfy $\texttt{MyNonce}(m, v)$.

**Example 2: $\mathcal{GL}(\mathsf{NSL})$.** The language $\mathcal{GL}(\mathsf{NSL})$ of the protocol NSL can be *identical* with the language $\mathcal{GL}(\mathsf{NS})$. After all, it too has initiator and responder roles, each of length three, with exactly the same parameters, as well as the listener role. One of the messages on the roles is different, but, since the parameters are the same, this necessitates no change in the language itself.

**Semantics of $\mathcal{GL}(\Pi)$.** Suppose that $\Pi$ is a protocol, with language $\mathcal{GL}(\Pi)$. We will assume that there is a pair of tables $\tau_r, \tau_p$ such that—for every role $\rho \in \Pi$ and integer $i$ up to the length of $\rho$—$\tau_r(\rho, i)$ gives the role position predicate for the $i$th position on instances of $\rho$. Thus, for instance, in NS, we have $\tau_r(\mathsf{Init}, 1)$ is `InitFst` and $\tau_r(\mathsf{Resp}, 2)$ is `RespScd`, etc. We assume that $\tau_r$ is injective.

Moreover, $\tau_p$ gives the parameter predicate for a particular role and parameter. For instance, in NS, $\tau_p(\mathsf{Init}, A)$ is `Self` and $\tau_p(\mathsf{Init}, N_a)$ is `MyNonce`. However, $\tau_p(\mathsf{Resp}, A)$ is `Peer` and $\tau_p(\mathsf{Resp}, N_a)$ is `YourNonce`. We do not assume that $\tau_p$ is injective, although we do assume that when $\tau_p(\rho, a) = \tau_p(\rho', b)$, if any strand can be regarded as an instance of both $\rho$ and $\rho'$, then it has the same instance for $a$ in $\rho$ in the first case as for $b$ in $\rho'$ in the second. This makes the definition of satisfaction (Defn. 4.2) unambiguous. The tables $\tau_r, \tau_p$ match roles and parameters of $\Pi$ with the vocabulary of $\mathcal{GL}(\Pi)$. The tables for of $\mathcal{GL}(\mathsf{NS})$ are shown in Table 2.

Before defining satisfaction, we introduce some helpful notation. Recall that $s \downarrow j$ is the $j^{\text{th}}$ node along strand $s$.

**Definition 4.1** *If $\rho \in \Pi$ is a role of protocol $\Pi$, then $\mathsf{instances}(\rho)$ is the set of instances of $\rho$, i.e.*

$$\mathsf{instances}(\rho) = \{\alpha(\rho) \colon \alpha \textit{ is a substitution}\}.$$

*The (larger) set of strands that agree with a member of $\mathsf{instances}(\rho)$ on the first $i$ nodes, in having the same directed message for each, is defined:*

$$\mathsf{instances}(\rho|_i) = \{s \colon \exists r \in \mathsf{instances}(\rho) \,.\, \forall j \leq i \,.\, \mathsf{dmsg}(s \downarrow j) = \mathsf{dmsg}(r \downarrow j)\}.$$

*So $\mathsf{instances}(\rho|_i)$ contains a strand if it is indistinguishable from a run of $\rho$ while only $i$ events have occurred.*

*If* $s \in \mathsf{instances}(\rho|_i)$*, then* $\mathsf{match}(s, \rho, i)$ *is the substitution that causes the first* $i$ *nodes of* $s$ *to match the first* $i$ *nodes of* $\rho$*; i.e.* $\mathsf{match}(s, \rho, i) = \alpha$ *iff* $\alpha$ *is the most general substitution (if any exists) such that*

$$\forall j \leq i \,.\, \mathsf{dmsg}(s \downarrow j) = \mathsf{dmsg}(r \downarrow j).$$

We can now define satisfaction completely traditionally in the manner of Tarski. Observe that we distinguish variable assignments $\eta \colon \mathsf{Var} \to \mathsf{nodes}(\mathcal{F}) \cup \mathsf{ALG}$, which assign values to variables, from substitutions $\alpha$, which are essentially homomorphisms from the message algebra $\mathsf{ALG}$ to itself. However, in $H \circ \eta$ these notions can be meaningfully composed, sending each variable not to a message value or node in the source of $H$, but rather the corresponding value in the target.

**Definition 4.2** *Suppose* $\mathcal{F} = \langle \mathcal{N}, \to_E, \preceq, \mathsf{unique}, \mathsf{non} \rangle$ *is a fragment, and* $\eta$ *is a map from variables to values which are either messages in the message algebra* $\mathsf{ALG}$ *or else nodes of* $\mathcal{F}$*.*

*If* $x$ *is a variable of* $\mathcal{GL}(\Pi)$*, then* $\eta(x)$ *is just the result of applying the map* $\eta$ *to the variable* $x$*. If* $t$ *is a compound term* $\mathtt{pk}(t')$*,* $\mathtt{sk}(t')$*, or* $\mathtt{inv}(t')$*, then* $\eta(t)$ *is* $\mathsf{pk}(\eta(t'))$*,* $\mathsf{privk}(\eta(t'))$*, or* $\eta(t')^{-1}$ *resp., when the latter is well-defined, and is undefined otherwise.*[1]

*We define satisfaction* $\mathcal{F}, \eta \models \phi$ *for compound formulas using the standard Tarski clauses. For atomic formulas, we stipulate the clauses in Table 3. Any atomic formula containing* $t$ *is false if* $\eta(t)$ *is undefined.*

The table $\tau_p$ is well formed only if all choices of $\rho, a, i$ satisfying Conditions 1–2 in the clause for `ParamPred` yield the same outcome.

Observe in this definition that the formulas satisfied for $\mathcal{F}, \eta$ depend only on the nodes within $\mathcal{F}$. What a strand would do "after" the part in $\mathcal{F}$ never changes the truth value of any atomic formula. Indeed:

**Lemma 4.3** *Let* $\phi$ *be a positive existential formula, and let* $H \colon \mathcal{F} \hookrightarrow \mathcal{E}$*. If* $\mathcal{F}, \eta \models \phi$ *then* $\mathcal{E}, (H \circ \eta) \models \phi$*.*

*Proof.* For the protocol-independent vocabulary, preservation of atomic formulas is immediate from the definition of homomorphism.

If $\mathcal{F}, \eta \models \mathtt{RolePosition}(t)$, then, letting $s \downarrow i = \eta(t)$, there is an $\alpha$ and an $r = \alpha(\rho)$ such that $\forall j \leq i \,.\, \mathsf{dmsg}(s \downarrow j) = \mathsf{dmsg}(r \downarrow j)$. Hence, letting $r' = H \circ \alpha(\rho)$, we have $\forall j \leq i \,.\, H(\mathsf{dmsg}(s \downarrow j)) = \mathsf{dmsg}(r' \downarrow j)$. So $\mathcal{E}, H \circ \eta \models \mathtt{RolePosition}(t)$.

If $\mathcal{F}, \eta \models \mathtt{ParamPred}(t, t')$, then there is a $\rho$ and a $a$ satisfying Clauses 1–3. Applying $H$, the clauses are also satisfied for $(H \circ \eta)(t) = H(s \downarrow i)$.

If satisfaction for $\Phi, \Psi$ is preserved under $H$, then so is satisfaction for $\Phi \wedge \Psi$, $\Phi \vee \Psi$, and $\exists x \,.\, \Phi$. □

[1] We use typewriter font for the syntactic constants, and sans serif for the functions in the models that interpret them.

| | |
|---|---|
| $\mathcal{F}, \eta \models t = t'$ | iff $\eta(t) = \eta(t')$; |
| $\mathcal{F}, \eta \models \texttt{Preceq}(t, t')$ | iff $\eta(t) \preceq \eta(t')$; |
| $\mathcal{F}, \eta \models \texttt{Coll}(t, t')$ | iff $\eta(t) \Rightarrow^* \eta(t')$ or $\eta(t') \Rightarrow^* \eta(t)$; |
| $\mathcal{F}, \eta \models \texttt{Unq}(t)$ | iff $\eta(t) \in \mathsf{unique}$; |
| $\mathcal{F}, \eta \models \texttt{Non}(t)$ | iff $\eta(t) \in \mathsf{non}$; |
| $\mathcal{F}, \eta \models \texttt{UnqAt}(t', t)$ | iff $\eta(t) \in \mathsf{unique}$ and $\eta(t') = s \downarrow i$ where $s \downarrow i \in \mathsf{nodes}(\mathcal{F})$ and $\eta(t)$ originates at $s \downarrow i$; |

$\mathcal{F}, \eta \models \texttt{RolePosition}(t)$ iff, letting $\tau_r(\rho, i) = \texttt{RolePosition}$, we have:

1. $\eta(t) = s \downarrow i$ where $s \downarrow i \in \mathsf{nodes}(\mathcal{F})$;
2. $s \in \mathsf{instances}(\rho|_i)$;

$\mathcal{F}, \eta \models \texttt{ParamPred}(t, t')$ iff, letting $\tau_p(\rho, a) = \texttt{ParamPred}$, we have:

1. $\eta(t) = s \downarrow i$ where $s \downarrow i \in \mathsf{nodes}(\mathcal{F})$;
2. $\mathsf{match}(s, \rho, i) = \alpha$
3. $\alpha(a) = \eta(t')$.

Table 3: Clauses for satisfaction

$\Phi$ *entails* $\Psi$ if for all $\Pi$-fragments $\mathcal{F}$, $\mathcal{F}, \eta \models \Phi$ implies $\mathcal{F}, \eta \models \Psi$. $\Phi$ and $\Psi$ are *equivalent* if each entails the other. Two variable assignments $\eta, \theta$ *agree on a set of variables* $\mathcal{V}$, written $\eta \sim_{\mathcal{V}} \theta$, if $\eta(x) = \theta(x)$ for every variable $x \in \mathcal{V}$. We write $\mathsf{fv}(\Phi)$ for the free variables of $\Phi$, and write $\eta \sim_\Phi \theta$ to mean $\eta \sim_{\mathsf{fv}(\Phi)} \theta$.

**Definition 4.4** *Let* $\Pi$ *be a protocol, and let* $G \in \mathcal{GL}(\Pi)$ *be a closed formula of the form (14).* $\Pi$ achieves *the goal* $G$ *if, for every* realized *fragment* $\mathcal{D}$, $\mathcal{D} \models G$.

# 5 Characteristic Fragments and Formulas

Each of the fragments in Figs. 8–14 is a structure for $\mathcal{GL}(\mathsf{NS})$. Looking at each one, we can read off a formula that characterizes it. For instance, for fragment $\mathcal{F}$ in 8, we had the formula Eqn 2.

This suggests the idea of a *characteristic formula*: A conjunction of atomic formulas $\Phi$ is a characteristic formula for $\mathcal{F}$ if $\Phi$ is a logically strongest conjunction of atoms that is true in $\mathcal{F}$. To define characteristic formulas, we start with the familiar logical notion of "diagram," relative to a variable assignment that covers the values relevant to $\mathcal{F}$.

**Definition 5.1** *Suppose* $\mathcal{F}$ *is a* $\Pi$*-fragment, and let* $S = \mathsf{nodes}(\mathcal{F}) \cup \mathsf{Params}(\mathcal{F})$ *be the set containing all its nodes and the message parameters to its strands. Let* $\eta$ *be a variable assignment which is injective and covers* $S$*, i.e.* $S \subseteq \mathrm{range}(\eta)$*, and let* $V$ *be the inverse image of* $S$ *under* $\eta$*. Let* $D$ *be the set of atomic formulas* $\phi$ *of* $\mathcal{GL}(\Pi)$ *such that* $\mathsf{fv}(\phi) \subseteq V$ *and* $\mathcal{F}, \eta \models \phi$.

*Because $V$ is finite, $D$ is finite.*
*The* $\eta$-diagram of $\mathcal{F}$, *written* $\delta_\eta(\mathcal{F})$, *is the conjunction* $\bigwedge D$.

Whenever we write $\delta_\eta(\mathcal{F})$, we assume that $\eta$ is injective and covers $S = \mathsf{nodes}(\mathcal{F}) \cup \mathsf{Params}(\mathcal{F})$. When $\eta$ is irrelevant, we omit it and write $\delta(\mathcal{F})$.

The diagram of $\mathcal{F}$ is a strongest formula that is true of $\mathcal{F}$. In particular, any other true atomic formula $\psi$ is a consequence of $\delta(\mathcal{F})$, as long as we specify the meanings of any additional variables in $\psi$:

**Lemma 5.2** *Let* $V = \mathsf{fv}(\delta_\eta(\mathcal{F}))$. *Suppose that* $\eta \sim_V \theta$, *and* $\mathcal{F}, \theta \models \psi$. *Then there is a finite set of equations* $t_i = y_i$ *where* $\mathsf{fv}(t_i) \subseteq V$ *and* $y_i \notin V$ *such that:*

1. $\mathcal{F}, \theta \models \bigwedge(t_i = y_i)$; *and*
2. $\delta_\eta(\mathcal{F}) \wedge \bigwedge(t_i = y_i)$ *entails* $\psi$.

*Proof.* If $\mathcal{F}, \theta \models \psi$, where $\psi$ is $R(s_1, \ldots, s_n)$, then $\langle \theta(s_1), \ldots, \theta(s_n) \rangle$ is in the extension of $R$ in $\mathcal{F}$. If $y \in \mathsf{fv}(s_i)$, then $\theta(y)$ is either a node, a parameter, or the result of applying a key function $f$ to a parameter $a$. In the first two cases, $\theta(y)$ is also in the range of $\eta$, say $\eta(x)$, so use the equation $x = y$. In the last case, $a$ is in the range of $\eta$, say $\eta(x)$, so use the equation $f(x) = y$. □

A characteristic formula is any formula equivalent to the diagram:

**Definition 5.3** *A conjunction of atomic formulas* $\Phi$ *is a* characteristic formula for $\mathcal{F}$ via $\eta$, *written* $\Phi \in \mathsf{cform}_\eta(\mathcal{F})$, *if* $\Phi$ *is equivalent to* $\delta_\eta(\mathcal{F})$.

As examples, each of the formulas we mentioned in Section 3 as giving the "content" of one of the fragments in Figs. 8–14 is a characteristic formula for it. This is the reason why we repeated the content of the characteristic formula for the left hand fragments $\mathcal{F}$ in the right hand fragments $\mathcal{E}$, so that the latter would be a self-contained characteristic formula. Using the definitions:

**Lemma 5.4** *If* $\Phi, \Psi \in \mathsf{cform}_\eta(\mathcal{E})$, *then* $\Phi$ *and* $\Psi$ *are equivalent.*
*If* $\Phi \in \mathsf{cform}_\eta(\mathcal{E})$ *and* $\Psi \in \mathsf{cform}_\theta(\mathcal{E})$, *then their existential closures* $\exists \overline{x} \,.\, \Phi$ *and* $\exists \overline{y} \,.\, \Psi$ *are equivalent.*

A characteristic formula may be quite a lot shorter than $\delta_\eta(\mathcal{F})$. It does not need to mention nodes on a strand earlier than the last one in $\mathsf{nodes}(\mathcal{F})$, nor say that they are collinear with it and precede it, nor repeat that their parameters agree with those on the later node of the same strand. Nor do we need to include both $\mathtt{Unq}(v)$ and $\mathtt{UnqAt}(n, v)$. We prefer the former when $n$ is an initial transmission node, since any ingredient of its message must originate there. Otherwise, the latter is more informative, so we use that.

Eqn. 2 and the fragment $\mathcal{F}$ on the left of Fig. 8 have a very strong relation. Any fragment that is a homomorphic image of $\mathcal{F}$ will satisfy Eqn. 2, by Lemma 4.3. Indeed, conversely, any fragment that satisfies Eqn. 2 will be a homomorphic image of $\mathcal{F}$. That is because it must have a responder strand (including all three nodes) with freshly chosen nonce and uncompromised peer,

to satisfy Eqn. 2. Given the third of these nodes, and its parameters, we know just how to build the homomorphism. We call $\mathcal{F}$ a *characteristic fragment* for Eqn. 2, because being a homomorphic image of $\mathcal{F}$ characterizes whether a fragment satisfies Eqn. 2.

**Definition 5.5** *$\mathcal{E}$ is a* characteristic fragment *for $\Phi$ via variable assignment $\eta$ iff for every $\mathcal{F}$ and assignment $\theta$,*

$$\mathcal{F}, \theta \models \Phi \text{ iff there exists a unique } H\colon \mathcal{E} \hookrightarrow \mathcal{F} \text{ such that } (H \circ \eta) \sim_\Phi \theta. \quad (15)$$

**Lemma 5.6** *If $\mathcal{E}$ and $\mathcal{F}$ are characteristic fragments for $\Phi$ via $\eta$ and $\theta$ resp., then there is an isomorphism $H\colon \mathcal{E} \hookrightarrow \mathcal{F}$ such that $H \circ \eta \sim_\Phi \theta$.*

*Proof.* Since $\mathcal{E}$ is a characteristic fragment and $\mathcal{F}, \theta \models \Phi$, there is a *homomorphism* $H$. However, since $\mathcal{F}$ is a characteristic fragment, there is a homomorphism $K\colon \mathcal{F} \hookrightarrow \mathcal{E}$. Since there is only one homomorphisms $\mathcal{E} \hookrightarrow \mathcal{E}$ satisfying Eqn. 15, and the identity is one, $K \circ H$ must also be the identity. So $H$ is an isomorphism. □

Because of Lemma 5.6, we can regard *characteristic fragment* as a partial function from conjunctions of atoms to fragments (to within isomorphism). We write $\mathsf{cfrag}_\eta(\Phi)$ for this partial function. When we say $\mathsf{cfrag}_\eta(\Phi) = \mathcal{E}$, we mean that it is well-defined, and has value $\mathcal{E}$.

Characteristic fragments and characteristic formulas are connected *à la* Galois: When $\mathcal{E}$ is a characteristic fragment for a formula $\Phi$, and $\Phi'$ is a characteristic formula for $\mathcal{E}$, then $\Phi$ and $\Phi'$ are equivalent. When $\Phi$ is a characteristic formula for $\mathcal{E}$, and $\mathcal{F}$ is a characteristic fragment for $\Phi$, then $\mathcal{F} \hookrightarrow \mathcal{E}$.[2] Taking a characteristic formula may discard some information, because $\mathcal{GL}(\Pi)$ is of limited expressiveness, but taking a characteristic fragment does not.

**Lemma 5.7** *1. If $\Phi \in \mathsf{cform}_\eta(\mathcal{F})$ and $\mathsf{cfrag}_\theta(\Phi) = \mathcal{E}$, then $\mathcal{E} \hookrightarrow \mathcal{F}$.*

*2. When $\mathsf{cfrag}_\eta(\Phi) = \mathcal{F}$ and $\Psi \in \mathsf{cform}_\eta(\mathcal{F})$, then $\Psi$ is equivalent to $\Phi$.*

*Proof.* 1. When $\Phi \in \mathsf{cform}_\eta(\mathcal{F})$, by the definition, $\mathcal{F}, \eta \models \Phi$. Hence if $\mathsf{cfrag}_\theta(\Phi) = \mathcal{E}$ is well-defined, the definition of characteristic fragment says that $H\colon \mathcal{E} \hookrightarrow \mathcal{F}$ exists, where $\eta \sim_\Phi (H \circ \theta)$.

2. (a) $\Psi$ entails $\Phi$, because the latter is a conjunction of atomic formulas all satisfied in $\mathcal{F}$, and Lemma 5.2 ensures that $\delta_\eta(\mathcal{F})$ entails each of them.

(b) Conversely, we claim $\Phi$ entails $\Psi$: Let $\psi$ be any conjunct of the latter. $\mathcal{F}, \eta \models \psi$ by the definition of $\mathsf{cform}$. Therefore $\mathcal{E}, H \circ \eta \models \psi$ whenever $H\colon \mathcal{F} \hookrightarrow \mathcal{E}$. However, these $\mathcal{E}, \theta$ are precisely the satisfying interpretations of $\Phi$, by the definition of $\mathsf{cfrag}$. Thus, $\psi$ is satisfied whenever $\Phi$ is. □

In clause 1, $\mathsf{cfrag}(\mathsf{cform}(\mathcal{F}))$ may be properly less informative than $\mathcal{F}$, in the sense that the homomorphism is not an isomorphism. To take the simplest case,

[2] Indeed, this map is injective on the nodes, so that $\mathcal{F} \hookrightarrow_{ni} \mathcal{E}$.

suppose that $\mathcal{F}$ contains only $\bullet \xleftarrow{\{|a|\}_K}$. Then $\mathsf{cform}(\mathcal{F})$ is just $\mathtt{Lsn}(n) \wedge \mathtt{Hear}(n, v)$, since $\mathcal{GL}(\Pi)$ has no way to say that $v$ is actually an encryption, rather than any other message. Thus, the characteristic fragment is $\bullet \xleftarrow{x}$, where $x$ is an indeterminate that can be replaced by any message. In particular, $\alpha = x \mapsto \{|a|\}_K$ determines the non-isomorphism from $\bullet \xleftarrow{x}$ to $\bullet \xleftarrow{\{|a|\}_K}$.

**Role-specific Formulas.** In fact, $\mathsf{cform}(\mathcal{F})$ yields a formula of a special, well-typed kind, ensuring that $\mathsf{cfrag}(\mathsf{cform}(\mathcal{F}))$ is actually well-defined. We call formulas like this *role-specific* formulas.

**Definition 5.8** *Let $\Phi$ be a conjunction of atomic formulas.*

*A variable $n \in \mathsf{fv}(\Phi)$ is a* node *variable in $\Phi$ if it occurs in some conjunct of $\Phi$ as the argument to a role position predicate* $\mathtt{RolePosition}(n)$.

*A variable $v \in \mathsf{fv}(\Phi)$ is a* message *variable in $\Phi$ if it occurs in the second argument of a parameter predicate* $\mathtt{ParamPred}(n, v)$.

*$\Phi$ is* role-specific *iff (i) its node variables and message variables partition $\mathsf{fv}(\Phi)$; (ii) only message variables appear as argument to a key function,* `Unq`*,* `Non`*, or in the second position of* `UnqAt`*; and (iii) only node variables appear (a) as arguments to* `Preceq` *or* `Coll`*, or (b) as the first argument to* `UnqAt` *or a parameter predicate.*

All of the characteristic formulas in Section 3 are role-specific.

**Lemma 5.9** *Let $\Phi$ be role-specific, with $\mathcal{F}, \eta \models \Phi$. If $x$ is a node variable of $\Phi$, then $\eta(x) \in \mathsf{nodes}(\mathcal{F})$. If $x$ is a message variable of $\Phi$, then $\eta(x) \in \mathsf{ALG}$.*

*For all skeletons $\mathbb{A}$, $\delta_\eta(\mathbb{A})$ is role-specific.*

*Proof.* 1. From the definitions.

2. If $\mathbb{A}$ is a skeleton, it has no adversary nodes. So every node $n \in \mathsf{nodes}(\mathbb{A})$ satisfies some role position predicate, which therefore appears in $\delta_\eta(\mathbb{A})$. Every parameter satisfies some parameter predicate with one of these nodes. By the construction of $\eta$ in Defn. 5.1, this partitions the variables in $\delta_\eta(\mathbb{A})$. The remaining conditions follow from the type constraints in the definition of satisfaction. □

$\delta_\eta(\mathcal{F})$ is typically not role-specific for non-skeletons $\mathcal{F}$. For instance, consider a fragment containing two adversary nodes $n, m$ transmitting basic values, with $n \preceq m$. Then $\delta_\eta(\mathcal{F})$ contains atomic formulas involving `Preceq`, but no role predicates that say what *regular* roles $m, n$ belong to.

**Lemma 5.10** *Let $\Phi$ be a role-specific conjunction of atomic formulas. $\mathcal{F}, \eta \models \Phi$ iff $\mathsf{skeleton}(\mathcal{F}), \eta \models \Phi$.*

*Proof.* From right to left, the implication holds because the identity is a homomorphism $\mathsf{skeleton}(\mathcal{F}) \hookrightarrow \mathcal{F}$.

From left to right, the implication holds because $\eta(x)$ is a regular node or a parameter to some regular node, for every $x \in \mathsf{fv}(\Phi)$. Thus, any fact involving $x$ is preserved in $\mathsf{skeleton}(\mathcal{F})$. □

We now prove that $\mathsf{cfrag}(\Phi)$ is well-defined for role-specific $\Phi$. However, to do so, we will need slightly adapted versions of the lemmas [37, Lemmas 3.14–3.15]. A map $f$ is *universal* in some set of maps $F$ if $f \in F$ and, for every $f' \in F$, there is exactly one $g$ such that $f'$ is of the form $f' = g \circ f$.

**Lemma 5.11** *Suppose that* $H \colon \mathcal{F} \hookrightarrow \mathcal{E}$.

1. *Suppose that* $H(a) = H(b)$ *for* $a, b \in \mathsf{ALG}$. *The set of homomorphisms* $\{K \colon \mathcal{F} \hookrightarrow \mathcal{E}' \colon K(a) = K(b)\}$ *has a universal member* $K_0$.
2. *Suppose that* $H(n) = H(m)$ *for* $n, m \in \mathsf{nodes}(\mathcal{E})$. *The set of homomorphisms* $\{K \colon \mathcal{F} \hookrightarrow \mathcal{E}' \colon K(n) = K(m)\}$ *has a universal member* $K_0$.
3. *Suppose that* $H(n) \preceq_{\mathcal{E}} H(m)$ *for* $n, m \in \mathsf{nodes}(\mathcal{E})$. *The set of homomorphisms* $\{K \colon \mathcal{F} \hookrightarrow \mathcal{E}' \colon K(n) \preceq_{\mathcal{E}'} K(m)\}$ *has a universal member* $K_0$.
4. *Suppose that* $H(a) \in \mathsf{unique}(\mathcal{E})$. *The set of homomorphisms* $\{K \colon \mathcal{F} \hookrightarrow \mathcal{E}' \colon K(a) \in \mathsf{unique}(\mathcal{E}')\}$ *has a universal member* $K_0$.

We will not re-prove this lemma here; adapting the proofs is routine.

**Theorem 5.12** *If a satisfiable conjunction* $\Phi$ *of atomic formulas is role-specific, then* $\mathsf{cfrag}_\eta(\Phi)$ *is well-defined, and is a skeleton, for some* $\eta$.

*Proof.* We will assume that $\Phi$ is in left-associated form $((\phi_1 \wedge \phi_2) \wedge \ldots) \wedge \phi_j$, and that the leftmost occurrence of a node variable is a role position predicate, and the leftmost occurrence of a message variable is a parameter predicate. We work by induction on $j$.

*Base case,* $j = 0$. In this case, $\Phi$ is the vacuously true empty conjunction, and its characteristic fragment is the empty fragment with $\mathsf{nodes} = \emptyset$, etc. There is in fact exactly one homomorphism from the empty fragment to any fragment, so it satisfies the condition for $cs(\Phi)$. Indeed, it is a skeleton. The variable assignment $\eta$ can be any injective assignment.

*Induction step.* Here we assume that $\mathsf{cfrag}_\eta(\Phi) = \mathbb{A}$ is well-defined, and a skeleton, and we consider the satisfiable, role-specific formula $\Phi \wedge \phi_j$. Since $\Phi \wedge \phi_j$ is satisfiable, there is a fragment $\mathcal{F}$ such that $\mathcal{F}, \theta \models \Phi \wedge \phi_j$. By Lemma 5.10, $\mathsf{skeleton}(\mathcal{F}), \theta \models \Phi \wedge \phi_j$. Since $\mathsf{cfrag}_\eta(\Phi) = \mathbb{A}$, there is a (unique) $H$ such that $H \colon \mathbb{A} \hookrightarrow \mathsf{skeleton}(\mathcal{F})$, and $\theta \sim_\Phi H \circ \eta$.

Since we have $H$, we may apply the clauses of Lemma 5.11 to $H$. The universality of the maps they guarantee is what ensures that their result is universal for the extended formula $\Phi \wedge \phi_j$. We take cases on the form of $\phi_j$:

$\phi_j$ **is** `Preceq`$(m, n)$**:** By role-specificity, $\eta(m)$ and $\eta(n)$ are nodes in $\mathbb{A}$. Thus, the desired skeleton $\mathbb{B}$ is the target of the homomorphism $K$ of clause 3.

$\phi_j$ **is** $\mathtt{Coll}(m,n)$**:** By role-specificity, $\eta(m)$ and $\eta(n)$ are nodes in $\mathbb{A}$, of the forms $s \downarrow k$ and $s' \downarrow \ell$. Assuming w.l.o.g. that $k \leq \ell$, apply clause 2 to $s \downarrow k$ and $s' \downarrow k$.

$\phi_j$ **is** $\mathtt{Unq}(t)$**:** Adding $\eta(t)$ to $\mathsf{unique}(\mathbb{A})$, and using clause 4 yields a universal result.

$\phi_j$ **is** $\mathtt{Non}(t)$**:** Adding $\eta(t)$ to $\mathsf{non}(\mathbb{A})$ yields the desired result.

$\phi_j$ **is** $\mathtt{UnqAt}(n,t)$**:** By role-specificity, $\eta(n)$ is a node in $\mathbb{A}$. Since $H(\eta(t)) \in \mathsf{unique}(\mathcal{F})$, it is a basic value. Let $p$ be a path to an occurrence of $H(\eta(t))$ within $\mathsf{msg}(H(\eta(n)))$ as an ingredient (i.e., not as an encryption key); and let $p'$ be the longest prefix of $p$ that is a path within $\eta(t)$. If $a$ is the value occurring at $p'$ in $\mathsf{msg}(\eta(n))$, $H(a) = H(\eta(t))$. Thus, we may apply clause 1.

$\phi_j$ **is** $s = t$**:** If $\eta(s), \eta(t) \in \mathsf{ALG}$, we apply clause 1. Otherwise, by role-specificity, they are both nodes in $\mathit{node}(\mathbb{A})$. Hence, we may apply clause 2.

$\phi_j$ **is** $\mathtt{RolePos}(n)$**:** If $n \in \mathsf{fv}(\Phi)$, then there is also an earlier conjunct $\mathtt{RolePos}'(n)$ by role-specificity. If $\mathtt{RolePos} = \tau_r(\rho, i)$, then $\mathtt{RolePos}' = \tau_r(\rho', i)$, and we can apply clause 2 $i$ times, once for each of the nodes up to $\eta(n)$.

If instead $n \notin \mathsf{fv}(\Phi)$, then we make a copy of $\rho$ instantiating its parameters with values not yet used in $\mathbb{A}$ to form strand $s$, and we put $\mathsf{nodes}(\mathbb{B}) = \mathsf{nodes}(\mathbb{A}) \cup \{s \downarrow k \colon k \leq i\}$. To form $\eta'$, we map $n \mapsto (s \downarrow i)$, and for $\mathsf{fv}(\Phi)$, $\eta'$ agrees with $\eta$.

$\phi_j$ **is** $\mathtt{ParamPred}(n,t)$**:** There is an earlier conjunct $\mathtt{RolePos}(n)$, by role specificity. Since $\Phi \wedge \phi_j$ is satisfiable, there are $\rho, i, a$ such that $\mathtt{RolePos} = \tau_r(\rho, i)$ and $\mathtt{ParamPred} = \tau_p(\rho, a)$. Let $v$ be the value of parameter $a$ in node $\eta(n)$.

If $t$ is a variable $x \notin \mathsf{fv}(\Phi)$, then leave $\mathbb{A}$ unchanged and let $\eta'$ be $\eta$ with $x \mapsto v$.

If $t \in \mathsf{fv}(\Phi)$ or if $t$ is $g(x)$ where $x \in \mathsf{fv}(\Phi)$ and $g$ is a key function, then apply clause 1 to $\eta(t)$ and $v$.

If $t$ is $g(x)$ where $x \notin \mathsf{fv}(\Phi)$, then let $\eta'$ be $\eta$ with $x \mapsto b$, where $b$ is a new parameter of sort principal name. Now apply clause 1 to $\eta'(t)$ and $v$.

Each inductive case also determines a variable assignment $\theta$ for the formula. □

By Lemma 5.9 and Thm. 5.12, we can always assume that $\mathsf{cfrag}(\Phi)$ is defined when $\Phi \in \mathsf{cform}(\mathbb{A})$. Even if $\Phi$ is not itself of the right syntactic form, it is equivalent to something of the right form, and we will always assume that a role-specific form has been chosen.

Goal formulas and families of homomorphisms are now equivalent now in the following sense, using "achievement" and "enforcement." Achievement means that the closed formula is true in every realized fragment; enforcement means

that every homomorphism to a realized skeleton factors through some member of the family (Defns. 3.2 and 4.4).

Our assumption in this theorem that each $\psi_i$ entails $\phi$ is not a significant restriction. Since $\phi$ is already available as a hypothesis in the goal formula, we could replace any $\psi_i$ that did not meet this assumption by $\phi \wedge \psi_i$ without changing the meaning of the formula. Indeed, in case $\psi_i$ is a characteristic formula, it is already of this form, as we illustrated in Section 3.

**Theorem 5.13** *Let* $\Pi$ *be a protocol and let* $G \in \mathcal{GL}(\Pi)$ *be a formula*

$$\forall \overline{x} \,.\, (\phi \quad \longrightarrow \quad (\exists \overline{y_1} \,.\, \psi_1) \vee (\exists \overline{y_2} \,.\, \psi_2) \vee \ldots \vee (\exists \overline{y_j} \,.\, \psi_j)),$$

*where* $\phi, \psi_i$ *are satisfiable and role-specific, and each* $\psi_i$ *entails* $\phi$*. Then there exists a family of homomorphisms*

$$H_i \colon \mathsf{cfrag}(\phi) \to \mathsf{cfrag}(\psi_i)$$

*based in* $\mathsf{cfrag}(\phi)$ *such that*

$$\Pi \textit{ achieves } G \textit{ iff } \Pi \textit{ enforces } \{H_i\}_{1 \leq i \leq j}.$$

*Proof.* By Thm. 5.12, $\mathsf{cfrag}_\eta(\phi)$ and each $\mathsf{cfrag}_{\theta_i}(\psi_i)$ is a well-defined skeleton. Since $\psi_i$ entails $\phi$, the latter is satisfied in $\mathsf{cfrag}_{\theta_i}(\psi_i)$. Since $\mathsf{cfrag}_\eta(\phi)$ is a characteristic fragment, there is a homomorphism $H_i \colon \mathsf{cfrag}_\eta(\phi) \to \mathsf{cfrag}_{\theta_i}(\psi_i)$ such that $\theta_i \sim_\phi H_i \circ \eta$. We use the family $\{H_i\}_{1 \leq i \leq j}$ of these homomorphisms.

1. Suppose that $\Pi$ achieves $G$ and $\mathcal{D}$ is a realized $\Pi$-fragment. We must show that if there is a homomorphism $K \colon \mathsf{cfrag}_\eta(\phi) \to \mathcal{D}$, then $K$ factors through one of the $H_i$.

   If $K \colon \mathsf{cfrag}_\eta(\phi) \to \mathcal{D}$, then $\mathcal{D}, (K \circ \eta) \models \phi$. Since $G$ is satisfied, one of the disjuncts $\exists \overline{y_i} \,.\, \psi_i$ must be satisfied too, i.e. $\mathcal{D}, (K \circ \eta) \models \exists \overline{y_i} \,.\, \psi_i$. Thus, for some $\zeta$ that differs from $K \circ \eta$ only on $\overline{y_i}$ we have $\mathcal{D}, \zeta \models \psi_i$. Since $\mathsf{cfrag}_{\theta_i}(\psi_i)$ is a characteristic fragment, we have the desired $J \colon \mathsf{cfrag}_{\theta_i}(\psi_i) \to \mathcal{D}$ such that $\zeta \sim_{\psi_i} J \circ H_i \circ \eta$.

2. Suppose that every homomorphism from $\mathsf{cfrag}_\eta(\phi)$ to a realized $\mathcal{D}$ factors through one of the $H_i$. We must show that $\mathcal{D}$ satisfies $G$.

   Suppose that $\zeta$ is a variable assignment. If $\mathcal{D}, \zeta \not\models \phi$, then $\mathcal{D}, \zeta \models \phi \longrightarrow \bigvee_i \psi_i$. So assume $\mathcal{D}, \zeta \models \phi$. By the definition of characteristic fragment, there is a $K \colon \mathsf{cfrag}_\eta(\phi) \to \mathcal{D}$ such that $\zeta \sim_\phi K \circ \eta$. Factoring $K = J \circ H_i$, we have $\zeta \sim_\phi J \circ (H_i \circ \eta)$. So $\zeta \sim_\phi J \circ \theta_i$, whence $\mathcal{D}, \zeta \models \exists \overline{y_i} \,.\, \psi_i$. □

A corollary, due originally to Ramsdell [53, Thm. 2], works in the opposite direction, constructing a formula from a family that $\Pi$ enforces:

**Corollary 5.14** *Let* $\{H_i \colon \mathbb{A} \to \mathcal{E}_i\}_{1 \leq i \leq j}$ *be a family of homomorphisms based in a skeleton* $\mathbb{A} = \mathsf{cfrag}_\eta(\phi)$*. Let* $\theta_i = H_i \circ \eta$*; let* $\psi_i = \mathsf{cform}_{\theta_i}(\mathcal{E}_i)$*; and let* $\overline{y_i} = \mathsf{fv}(\psi_i) \setminus \mathsf{fv}(\phi)$*.*

*If* $\Pi$ *enforces* $\{H_i\}_{1\leq i\leq j}$*, then* $\Pi$ *achieves goal formula*

$$\phi \longrightarrow \bigvee_{1\leq i\leq j} \exists \overline{y_i} \,.\, \psi_i. \tag{16}$$

*Proof.* Applying Thm. 5.13 to Formula 16 as $G$, we generate a family

$$\{K_i \colon \mathsf{cfrag}_\eta(\phi) \hookrightarrow \mathsf{cfrag}_{\theta_i}(\psi_i)\}_{1\leq i\leq j};$$

Thm. 5.13 tells us that it suffices for us to show that $\Pi$ enforces $\{K_i\}_{1\leq i\leq j}$. By Lemma 5.7, for each $i$ there is an $L_i$ such that $L_i \colon \mathsf{cfrag}_{\theta_i}(\psi_i) \hookrightarrow \mathcal{E}_i$. Moreover, by the uniqueness in the definition of characteristic fragment, $H_i = L_i \circ K_i$.

Suppose now that $\mathcal{D}$ is realized and $M \colon \mathbb{A} \hookrightarrow \mathcal{D}$. Since $\Pi$ enforces $\{H_i\}_{1\leq i\leq j}$, $M$ factors through some $H_i$, i.e. $M = J \circ H_i$. But now $M = J \circ (L_i \circ K_i)$, so by associativity $M$ factors through $K_i$. So $\Pi$ enforces $\{K_i\}_{1\leq i\leq j}$. □

In Thm. 5.13 and Cor. 5.14, the finiteness of the index set $I = \{i \colon 1 \leq i \leq j\}$ is fundamentally irrelevant. If we enrich $\mathcal{GL}(\Pi)$ to allow infinitary disjunctions, then the corresponding results hold for infinite $I$ also, by the same arguments.

# 6 Enrich-by-Need Protocol Analysis

Thm. 5.13 and Cor. 5.14 suggest an approach to protocol analysis, which is in fact a deepening or formalization of the enrich-by-need idea.

Suppose we are interested in security goals that $\Pi$ achieves, using a particular premise $\phi$. First, we check that $\phi$ is role-specific. Section 3's examples suggest this will normally be the case. Working from $\mathsf{cfrag}(\phi) = \mathbb{A}$ as a starting point, we look for a family $\{H_i\}_{1\leq i\leq j}$ based in $\mathbb{A}$. When we find one, we take the targets $\mathcal{E}_i$ of the $H_i$. Their characteristic formulas $\psi_i$ determine a formula $G$ as in Eqn. 16. If $\Pi$ enforces $\{H_i\}_{1\leq i\leq j}$, then $\Pi$ achieves the formula $G$.

Suppose also that all of the $\mathcal{E}_i$ are realized. Let $G'$ be another formula of the same form, but using the formulas $\chi_1, \ldots, \chi_k$ to produce the conclusion $\bigvee_i \exists \overline{z_i} \chi_i$. If $\bigvee_i \exists \overline{y_i} \psi_i$ entails $\bigvee_i \exists \overline{z_i} \chi_i$, then $G'$ follows from the goal $G$ that we discovered. Otherwise, suppose $\bigvee_i \exists \overline{y_i} \psi_i$ does not entail $\bigvee_i \exists \overline{z_i} \chi_i$. Because the $\psi_i$ are logically strongest formulas true in the $\mathcal{E}_i$, it follows that $\bigvee_i \exists \overline{z_i} \chi_i$ is not true in one of the $\mathcal{E}_i$. Thus, that $\mathcal{E}_i$ is a counterexample to the goal $G'$ if $G'$ is not entailed by $G$.

Hence, a family enforced by $\Pi$ with realized targets $\mathcal{E}_i$ gives us a *strongest* goal. It dominates any other goal $G'$ with the same premise $\phi$.

It is also interesting to consider families where the targets may not be realized. If $\{H_j\}_{j\in J}$ has an unrealized member $H_i \colon \mathcal{F} \hookrightarrow \mathcal{E}_i$, and $\{K_j\}_{j\in J'}$ is a family based in $\mathcal{E}_i$, then

$$\{H_j\}_{j\in(J\setminus\{i\})} \cup \{K_j \circ H_i\}_{j\in J'}$$

is a family based in $\mathcal{F}$. It is indexed by $(J \setminus \{i\}) \cup J'$, if $J, J'$ are disjoint. Moreover, if $\Pi$ enforces $\{H_j\}_{j\in J}$ and $\{K_j\}_{j\in J'}$, then it also enforces $\{H_j\}_{j\in J\setminus\{i\}} \cup$

$\{K_j \circ H_i\}_{j \in J'}$. This is a form of compositionality, and—if suitable small steps $\{K_j\}_{j \in J'}$ can be found—offers a progressive way to compute families that eventually yield realized targets. We will call a small step like $\{K_j\}_{j \in J'}$ a *cohort*.

There are two main approaches to generating cohorts. The first introduces adversary strands. The second avoids them entirely, and works only with skeletons. To formulate the two approaches, we will use the notion of a *component*:

**Definition 6.1** *Suppose that a message $t_0$ is not a pair, but either a basic value, an encryption, a digital signature, or a hash. In keeping with our convention of representing digital signatures and hashes via encryptions, we will assume $t_0$ is either a basic value or an encryption.*

*Then $t_0$ is a* component *of a message $t_1$ iff either (i) $t_0 = t_1$, or else (ii) there exist $t_2, t_3$ such that $t_1 = t_2 \hat{} t_3$ and $t_0$ is (recursively) a component of either $t_2$ or $t_3$.*

So the components of $t_1$ result from it by unpairing until we reach non-pairs. Components are the important ingredients in messages, since an adversary with the right components can always pair and unpair to build the desired messages.

**Direct backward search.** One method, pursued by Athena and Scyther [58, 17], and related to NPA [46], is to consider, for each component in a reception node, which nodes could have transmitted it previously.

Suppose $n_1 \in \mathcal{F}$ is a reception node, and $c$ is a component of $\mathsf{msg}(n_1)$. If $c$ is not a component of any transmission node $n_0 \in \mathcal{F}$ with $n_0 \preceq_{\mathcal{F}} n_1$, then $\mathcal{F}$ cannot possibly be realized. In fact, not fragment that differs from $\mathcal{F}$ by adding 0 or more adversary pairing and unpairing strands can be realized.

Thus, the pair $n_1, c$ indicates a *problem* in $\mathcal{F}$ that must be solved by adding some other kind of information before $\mathcal{F}$ can become realized. This problem has several kinds of possible solution, in which we would

1. Apply a substitution $\alpha$ to $\mathcal{F}$, which unifies $c$ with some component transmitted earlier;
2. Add an instance of a protocol role, transmitting the component $c$;
3. Add an adversary encryption strand that transmits $c$, if it is an encryption; or
4. Add an adversary decryption strand that transmits $c$ by decrypting it (or a pair of which it is a component) from a larger encrypted unit $e$. This is relevant only when $e$ is an ingredient in some message transmitted on a regular node of some $\Pi$-strand [50, 13, 39].

These four groups of possibilities together cover the ways that $\mathcal{F}$ can be enriched to solve the problem $n_1, c$. We call this method "direct backward search" because—for each component $c$ received—it searches for transmission nodes earlier in time that would have sent $c$. It is thus directly motivated by the idea that every component received must previously have been sent.

**Authentication Test Search.** An alternative way to generate information-increasing steps is the authentication test method [39, 23].

Here also we consider a basic value or encryption $c$ such that $c \sqsubseteq \mathsf{msg}(n_1)$ for some reception node $n_1 \in \mathcal{F}$. We also choose a set of encryptions $S$. Suppose:

- If $\{|t|\}_K \in S$, then $K^{-1}$ is not a component of any node $m \preceq_{\mathcal{F}} n_1$.
- If $n_0 \prec_{\mathcal{F}} n_1$, then any path within $\mathsf{msg}(n_0)$ that leads to an occurrence of $c$ either traverses a member of $S$ or enters the key of an encryption. (In this case, we say that $c$ is found only within $S$ in nodes before $n_1$.)
- There is a path to an occurrence of $c$ within $\mathsf{msg}(n_1)$ that traverses no member of $S$ and enters the key in no encryption. (We say that $c$ is found outside $S$ in $n_1$.)

When these conditions hold, $n_1, c, S$ is an *unsolved authentication test* in $\mathcal{F}$. The "test" here is to explain how $c$ got outside of the encryptions $S$, as it somehow did, so as to be received as it was in $\mathsf{msg}(n_1)$. $S$ here may be the empty set, in which case the test is to explain how $c$ was transmitted at all. We call $S$ an *escape set*, since the test is to explain how $c$ has escaped from the encryptions in $S$.

To solve an unsolved test $n_1, c, S$, we may enrich the fragment $\mathcal{F}$ in the following ways:

1. If there is a substitution $\alpha$ such that $\alpha(c)$ is found only within $\alpha(S)$ in $\alpha(\mathsf{msg}(n_1))$, then applying $\alpha$ makes the test disappear.
2. Adding a listener node for a key $K^{-1}$ such that $\{|t|\}_K \in S$ to $\mathcal{F}$ explains $c$'s escape, as the adversary can hear $K^{-1}$ and use it to decrypt $\{|t|\}_K$.
3. Adding a regular transmission node $m_1$ in which $c$ is found outside $S$ explains $c$'s escape also, if $c$ was found only within $S$ in all earlier nodes $m_0 \Rightarrow^+ m_1$ on the same strand.

See [37] for a more precise description and various examples, and [54] to see how this idea is implemented in the protocol tool CPSA. In this method, the problems are the unsolved authentication tests.

**Cohorts.** Whether implemented by direct backward search or by the authentication test method, enrich-by-need protocol analysis turns on the notion of a cohort. Given a problem $\ell$ in $\mathcal{F}$, i.e. either $\ell$ is a pair $n_1, c$, where $c$ is a component received on $n_1$ but not transmitted earlier, or $\ell$ is an unsolved test $n_1, c, S$, a *cohort* for $\ell$ is a family of homomorphisms $\{H_i\}_{i \in I}$ based in $\mathcal{F}$ such that:

- If $H_i \colon \mathcal{F} \rightarrowtail \mathcal{E}_i$, then the image of $\ell$ is solved in $\mathcal{E}_i$.
- If $K \colon \mathcal{F} \rightarrowtail \mathcal{D}$, where $\mathcal{D}$ is realized, then $K = J \circ H_i$ for some $J$ and $i \in I$.

A cohort for $\ell$ is thus a set of maximally general ways of solving $\ell$.

As a special case, a cohort for $\ell$ could be the empty set of homomorphisms. The two approaches each suggest a way of computing cohorts, and these methods can return the empty set. When this happens, we have learnt that there are no realized fragments $\mathcal{D}$ accessible from $\mathcal{F}$. We express this by saying that $\mathcal{F}$ is *dead*: No homomorphism can lead to a realized skeleton.

**Search via Cohorts.** The cohort idea immediately explains enrich-by need as a form of search. Starting from an initial fragment, generally a skeleton $\mathbb{A}$, we look for a problem $\ell$ in it. If there is none, we can immediately construct a realized fragment from it. Otherwise, we choose a problem $\ell$ and construct the cohort for it. We build a directed graph rooted at $\mathbb{A}$ using the homomorphisms making up this cohort as edges.

At any stage, we choose a fragment on the fringe of the directed graph. If it is realized, we need not consider it further. If we find a problem and construct a cohort, we add those edges to the graph. If this cohort is empty, we mark the fragment as dead. Otherwise, the cohort helps us extend the fringe.

In some cases, the search terminates with a graph in which the fringe consists only of dead and realized fragments. Then the paths from $\mathbb{A}$ to realized fragments determine a family of homomorphisms based in $\mathbb{A}$. We can apply Thm. 5.13 to this family to decide whether a goal $G$ is achieved. We may also apply Cor. 5.14 to construct a strongest goal formula for the starting point $\mathbb{A}$.

If the search does not terminate, we may still falsify a goal $G$, if a path leads to a counterexample to the conclusion of $G$. Observe that as long as cohort generation is recursively enumerable, the existence of a counterexample to a goal $G$ with role-specific premise $\phi$ is recursively enumerable. Thus, the relation $\Pi$ *achieves* $G$ is co-r.e.

**Axiomatizing Enrich-by-need.** Homomorphisms determine a preorder, not a partial order, since $J \circ H$ is not always an isomorphism when $\mathcal{F} \xrightarrow{H} \mathcal{E} \xrightarrow{J} \mathcal{F}$. However, if $H, J$ map distinct nodes of their sources injectively to distinct nodes of their targets, then $J \circ H$ is an isomorphism. These *node-injective* homomorphisms $H \colon \mathcal{F} \rightarrowtail_{ni} \mathcal{E}$ determine a partial order $\leq_{ni}$ on skeletons to within isomorphism.

**Lemma 6.2 ([37, Lemma 3.11])** *$\leq_{ni}$ is a well-founded partial order. Indeed, for every $\mathcal{E}$, there are only finitely many non-isomorphic $\mathcal{F}$ such that $\mathcal{F} \leq_{ni} \mathcal{E}$.*

*When $\mathcal{F} \leq_{ni} \mathcal{E} \leq_{ni} \mathcal{F}$, $\mathcal{F}$ and $\mathcal{E}$ are isomorphic.*

Enrich-by-need protocol analysis is a search through part of the preorder $\rightarrowtail$. Skeletons $\mathbb{A}_0$ determine starting points for the search; protocol analysis then seeks realized fragments $\mathcal{D}$ such that $\mathbb{A} \rightarrowtail \mathcal{D}$. Both CPSA and Scyther implement this search. Scyther computes a set of fragments which have a minimality property [18]. CPSA computes a set of representative realized skeletons we call *shapes* [54]. Within the set of all realized $\mathbb{B}$ such that $\mathbb{A} \rightarrowtail \mathbb{B}$, the shapes are the *minimal* ones in the node-injective ordering $\leq_{ni}$ [23]. CPSA's test-and-solution

steps form a labeled transition system, where $\mathbb{A}_0 \overset{\ell}{\rightsquigarrow} \mathbb{A}_1$ means that $\mathbb{A}_0$ has an unsolved test described by the label $\ell$, and $\mathbb{A}_1$ contains one solution to this test. The LTS $\rightsquigarrow$ is a subrelation of $\mapsto$. Indeed, most of the search process works in the partial order $\leq_{ni}$. Although CPSA's implementation is somewhat different, its search could be separated into two phases. After an initial non-node-injective step, all of its test-solving could take place in the node-injective ordering (see [37, Thm. 6.5]).

Rather than specialize our results for CPSA, or Scyther, we *axiomatize* the crucial properties of problem-and-solution LTSs. This has an additional advantage. Namely, we can choose very small LTSs; they need only be large enough to model the steps in a single enrich-by-need search. These are often finite, indeed, often very small, LTSs.

We let $S$ be a set of fragments and $\Lambda$ be a set of labels. When modeling CPSA, a typical label $\ell \in \Lambda$ is of the form $n_1, c, S$, defining an unsolved authentication test. When modeling Scyther, typical $\ell$ take the form $n_1, c$, representing a component $c$ that is received without having previously been sent as a component. The transition relation implements the cohorts. For every $\ell$ of these forms, $\{\mathcal{E}_i \colon \mathcal{F} \overset{\ell}{\rightsquigarrow} \mathcal{E}_i\}$ defines the cohort solving the problem $\ell$. Strictly speaking, the homomorphisms, not their targets $\mathcal{E}_i$ are the members of the cohorts. However, in practice, $H_i$ is recoverable from the triple $\mathcal{E}, \ell, \mathcal{E}_i$. Thus, we will freely allow ourselves to pass from these triples to the homomorphisms themselves.

We also include one special value $\mathsf{dead} \in \Lambda$. When the cohort for $\mathcal{F}$ and a particular problem is empty, so that $\mathcal{F}$ is dead, we will in fact write $\mathcal{F} \overset{\mathsf{dead}}{\rightsquigarrow} \mathcal{F}$. Thus, a dead fragment stutters, and only a realized fragment is a terminal node in our transition system.

**Definition 6.3** *Suppose given $S$, a set of fragments, $\mathsf{dead} \in \Lambda$, and a ternary relation $\cdot \dot{\rightsquigarrow} \cdot \subseteq S \times \Lambda \times S$ such that $\mathcal{F} \rightsquigarrow \mathcal{E}$ implies $\mathcal{F} \mapsto \mathcal{E}$.*

*$(S, \Lambda, \rightsquigarrow)$ is a* problem-solution lts *or* PSLTS *iff:*

1. *If $\mathcal{F} \in S$, then $\mathcal{F}$ is realized iff there is no $\mathcal{E}$ such that $\mathcal{F} \rightsquigarrow \mathcal{E}$;*
2. *If $\mathcal{F} \overset{\mathsf{dead}}{\rightsquigarrow} \mathcal{E}$, then $\mathcal{F} = \mathcal{E}$ and there is no realized $\mathcal{D}$ such that $\mathcal{F} \mapsto \mathcal{D}$;*
3. *If $J \colon \mathcal{F} \mapsto \mathcal{D}$ from an unrealized $\mathcal{F}$ to a realized $\mathcal{D}$, then:*
   (a) *if $\mathcal{F} \overset{\ell}{\rightsquigarrow} \cdot$, then there exists $\mathcal{E}'$ s.t. $\mathcal{F} \overset{\ell}{\rightsquigarrow} \mathcal{E}'$, and $J = \mathcal{F} \overset{H}{\mapsto} \mathcal{E}' \overset{K}{\mapsto} \mathcal{D}$;*
   (b) *if $\mathcal{F} = \mathcal{F}_0 \rightsquigarrow \cdots \rightsquigarrow \mathcal{F}_i \rightsquigarrow \cdots$ is an infinite $\rightsquigarrow$-path, then for some $i$, $\mathcal{F}_i \not\mapsto \mathcal{D}$.*

*Let $S(\rightsquigarrow) = \{\mathcal{F} \colon \exists \mathcal{E} \,.\, \mathcal{F} \rightsquigarrow \mathcal{E}\} \cup \{\mathcal{E} \colon \exists \mathcal{F} \,.\, \mathcal{F} \rightsquigarrow \mathcal{E}\}$.*

We tacitly assume (e.g. in 3a) that the homomorphism $H$ can be recovered from $\mathcal{F} \overset{\ell}{\rightsquigarrow} \mathcal{E}$. Thus, when $\rightsquigarrow$ is iterated and we have $\mathcal{F} \overset{\sigma}{\rightsquigarrow}{}^* \mathcal{E}$ for some sequence of labels $\sigma$, then there is a definite $H \colon \mathcal{F} \mapsto \mathcal{E}$ corresponding to $\sigma$.

We may write $H_{\mathcal{F} \overset{\ell}{\rightsquigarrow} \mathcal{E}}$ for the homomorphism recovered from $\mathcal{F} \overset{\ell}{\rightsquigarrow} \mathcal{E}$, or $H_\sigma$ for the homomorphism for the path $\sigma = \mathcal{F}_0 \overset{\ell_1}{\rightsquigarrow} \cdots\cdots \overset{\ell_k}{\rightsquigarrow} \mathcal{F}_k$.

There is great freedom in defining PSLTSs. For one thing, we have mentioned that CPSA and Scyther construct them using different ideas [54, 17]. Moreover, a PSLTS may cover a very small finite set $S$ of fragments; in particular, it may cover only a particular starting point and the fragments we traverse to reach realized fragments. In simple cases, with CPSA, this may be just a few, or in complicated cases a hundred or two hundred.

**Theorem 6.4** *Suppose that $\cdot \stackrel{\cdot}{\rightsquigarrow} \cdot$ is a* PSLTS*, and $\mathcal{F}_0 \in S(\rightsquigarrow)$. If $\mathcal{F}_0 \mapsto_{ni} \mathcal{D}$ where $\mathcal{D}$ is realized, then there exists a realized $\mathcal{E}$ such that $\mathcal{F}_0 \rightsquigarrow^* \mathcal{E}$ and $\mathcal{E} \mapsto_{ni} \mathcal{D}$.*

*Proof.* Let $\Xi$ be the set of all sequences

$$\mathcal{F}_0 \stackrel{\ell_1}{\rightsquigarrow} \mathcal{F}_1 \rightsquigarrow \cdots \rightsquigarrow \mathcal{F}_{k-1} \stackrel{\ell_k}{\rightsquigarrow} \mathcal{F}_k$$

starting at $\mathcal{F}_0$ such that for each $i \leq k$, $\mathcal{F}_i \mapsto \mathcal{D}$. $\Xi \neq \emptyset$ because it contains the trivial sequence $\mathcal{F}_0$. By Def. 6.3, Clause 3b, there are maximal sequences in $\Xi$. So let $p = \mathcal{F}_0 \stackrel{\ell_1}{\rightsquigarrow} \mathcal{F}_1 \rightsquigarrow \cdots \rightsquigarrow \mathcal{F}_{k-1} \stackrel{\ell_k}{\rightsquigarrow} \mathcal{F}_k$ be maximal.

We claim that $\mathcal{F}_k$ is realized. Otherwise, there must be some transition $\mathcal{F}_k \stackrel{\ell}{\rightsquigarrow} \mathcal{C}$. Moreover, $\ell \neq \mathsf{dead}$, since then there is no realized fragment $\mathcal{B}$ such that $\mathcal{F}_k \mapsto \mathcal{B}$, so $\mathcal{F}_k \not\mapsto \mathcal{D}$, contradicting $p \in \Xi$.

Thus, if $\mathcal{F}_k$ is unrealized, there is a non-$\mathsf{dead}$ label $\ell$ such that $\mathcal{F}_k \stackrel{\ell}{\rightsquigarrow} \cdot$. Hence, by clause 3a, there exists $\mathcal{E}'$ s.t. $\mathcal{F}_k \stackrel{\ell}{\rightsquigarrow} \mathcal{E}'$, and $\mathcal{F} \mapsto \mathcal{E}' \mapsto \mathcal{D}$. Hence, $p \stackrel{\ell}{\rightsquigarrow} \mathcal{E}' \in \Xi$, contradicting the maximality of $p$. □

Combining this with Thm. 5.13, we have:

**Corollary 6.5** *Let $\phi$ be role specific, and let $G$ be $\forall \overline{x} \,.\, (\phi \longrightarrow \bigvee_i \exists \overline{y_i} \,.\, \psi_i)$. Let $\cdot \stackrel{\cdot}{\rightsquigarrow} \cdot$ be a* PSLTS*, and $\mathsf{cfrag}_\eta(\phi) \in S(\rightsquigarrow)$. Then $\Pi$ achieves $G$ iff, for every sufficiently long path*

$$\sigma = \mathsf{cfrag}_\eta(\phi) \stackrel{\ell_1}{\rightsquigarrow} \cdots\cdots \stackrel{\ell_k}{\rightsquigarrow} \mathcal{E}_k,$$

*either (i) $\ell_k = \mathsf{dead}$, or else (ii) $\mathcal{E}_k$ is realized and, for some $i$,*

$$\mathcal{E}_k, (H_\sigma \circ \eta) \models \exists \overline{y_i} \,.\, \psi_i.$$

*Proof.* For each realized $\mathcal{D}$, let $\sigma_\mathcal{D}$ be a path $\mathsf{cfrag}_\eta(\phi) \rightsquigarrow^* \mathcal{E}$ where $\mathcal{E}$ is realized and $\mathsf{cfrag}_\eta(\phi) \mapsto \mathcal{E} \mapsto \mathcal{D}$. Apply Thm. 5.13 to the (possibly infinite) family $\{H_{\sigma_\mathcal{D}}\}_\mathcal{D}$. □

# 7 Protocol Transformations

Protocol design is an art of reuse. A few basic patterns for achieving authentication and confidentiality—despite actively malicious parties—are frequently

adapted to new contexts. Designers combine these patterns, piggy-backing values on top of them, to solve many problems. The transformations modify message structure; add new transmissions or receptions on a given role; and add entirely new roles. Constructing protocols may be difficult, particularly for interactions involving more than two participants: Some data values may be shared among subsets of the participants, while remaining hidden from the other participants. Designers use existing protocols as heuristics for parts of the protocol, welding the parts cleverly together, so that the transformed protocol preserves the goals achieved by the components, while achieving additional goals.

Our goal here is not to make this cleverness unnecessary, but to explain it semantically. Thm. 7.12 justifies inferring that a transformed protocol satisfies some security goals, when the source protocol did. Although a logical result about models of protocol behavior and the formulas they satisfy, it is a corollary of a logic-free theorem (Thm. 7.8). The latter concerns only fragments, homomorphisms between them, and PSLTSs for source protocol and transformed protocol. These PSLTSs formalize relations between the activity of protocol analysis in the two protocols.

We start by considering a number of examples, which motivate a definition of protocol transformation. This is a rather inclusive notion, which includes many unsound transformations. Its technical motivation is that transformations as defined here form a full and faithful functor, transforming skeletons and homomorphisms of the source protocol to skeletons and transformations of the target protocol.

In this regard, skeletons—having no adversary behavior—are more canonical than fragments in general. It would require additional machinery to extend the methods given here to fragments in general. It is hard to translate the adversary strands except by the syntactic forms of the messages they send and receive. By contrast, a regular strand $s$ can be identified by giving a role $\rho$ that it instantiates, and the substitution $\alpha$ such that $s = \alpha(\rho)$, as in Defn. 7.3. We will thus concentrate on skeletons, leaving it to future work with greater focus on syntactic forms to integrate the remaining fragments into our picture.

We start first with some example protocol transformations, introducing a definition (Section 7.1). We then prove that transformations yield full and faithful functors on skeletons (Section 7.2). Finally, in Section 7.4 we give Thm. 7.8, generalizing Thm. 6.4. The main theorem Thm. 7.12 then follows.

## 7.1 Some Protocol Transformations

We start by recalling the protocol HD, defined in Fig. 1. Its security goals may be expressed in a language $\mathcal{GL}(\mathsf{HD})$, as described in Section 4. We do this using the standard protocol-independent vocabulary, and the protocol-specific predicates shown in Table 4.

We mentioned two transformations where HD is the source protocol and NS is the target protocol. First, we can associate the HD initiator with the first two nodes of the NS initiator, letting the nonce $N$ represent the NS initiator's nonce $N_a$. In this transformation, we associate the HD responder with the NS

responder, which receives an encrypted form of the initiator's nonce on its first node, and retransmits that value outside that encryption in its second node.

Alternatively, we can associate the HD initiator with the second and third nodes of the NS responder, letting the nonce $N$ represent the NS responder's nonce $N_b$. Now, we will associate the HD responder with the NS initiator, which receives an encrypted form of the responder's nonce on its second node, and retransmits that value outside that encryption in its third node. The HD responder receives the nonce $N$, and this is associated with the nonce $N_b$ received by the NS initiator.

This suggests that a protocol transformation $T$ with source protocol $\Pi_1$ and target protocol $\Pi_2$ should consist of a map that:

- sends each source protocol role $\rho_1 \in \Pi_1$ to a target protocol role $\rho_2 \in \Pi_2$;
- associates each node along role $\rho_1$ with a node along $\rho_2$; and
- sends each parameter $a$ of role $\rho_1$ to a parameter $b$ of $\rho_2$.

**The Yes-or-No Protocol.** The Yes-or-No Protocol YN allows a Questioner to ask a question $Q$, to which the Answerer gives a private, authenticated reply; YN is constructed by two transformations of HD. In YN, the question and answer should each remain secret. Indeed, the protocol should prevent even an adversary who has guessed the question from determining what answer was given. The Questioner authenticates the Answerer as supplying an answer.

The Questioner chooses two random nonces, and encrypts them, together with the question. The Answerer releases the first of the two nonces $Y$ to indicate a *yes*, and the second $N$ to indicate a *no*. No adversary learns anything, since whichever nonce was released, the questioner was equally likely to have used it in the other position.

The protocol has four roles (Fig. 15). One describes the behavior of a Questioner receiving an affirmative answer. The second describes the behavior of a Questioner receiving a negative answer. The remaining two describe the behavior of an Answerer providing an affirmative and respectively negative answer. Which of the two nonces has been released tells the Questioner which reply the Answerer has made.

We summarize $\mathcal{GL}$(YN) in Table 5.

The protocol is interesting partly because it is an example of a protocol that exhibits branching behaviors. Any instance of the first node of the QAf role is also an instance of the first node of the QNg role. A partial execution which

| $\tau_r$ | 1 | 2 |
|---|---|---|
| Init | `InitFst` | `InitScd` |
| Resp | `RespFst` | `RespScd` |

| $\tau_p$ | $B$ | $N$ |
|---|---|---|
| Init | `Peer` | `MyNonce` |
| Resp | `Self` | `YourNonce` |

Table 4: $\tau_r$ and $\tau_p$ for $\mathcal{GL}$(HD)

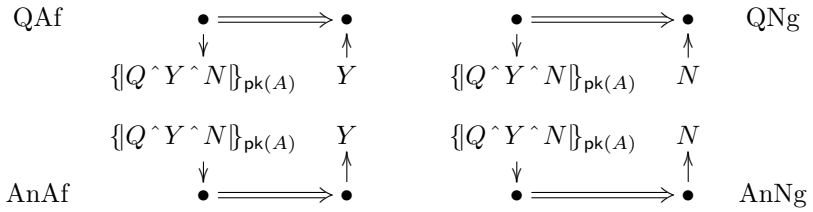



Figure 15: The Yes-or-No Protocol YN

has only reached this step is of both forms. The same is true of the roles AnAf and AfNg.

We can view either half of this diagram as a transformation of the protocol HD. In transforming HD to the left (affirmative) half of YN, we send the initiator role to QAf, and the responder role to AnAf. The HD nonce $N$ will be associated with the affirmative YN nonce $Y$. The principal name $B$ is associated with $A$.

In transforming HD to the right (negative) half of YN, we send the initiator role to QNg, and the responder role to AnNg. The HD nonce $N$ is now associated with the affirmative YN nonce $Y$, and $B$ is associated with $A$.

**Mutually Authenticated Yes-or-No Protocol.** As a final example, we present a mutually authenticated version of YN. Here, the Answerer starts by giving the Questioner a nonce $R$ to use when asking a question. The presence of this $R$ identifies the question $Q$ as originating with $B$; it is a sort of ticket enabling $B$ to ask a question of $A$. Perhaps $A$ will charge for the service.

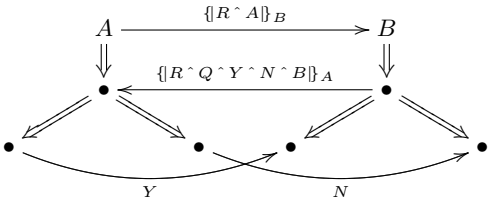



Figure 16: Answering Questions with $\mathsf{YN}^+$

There is a transformation from YN to $\mathsf{YN}^+$ in which we map AnAf and AnNg to the two roles shown on the left side and QAf and QNg to the two on the right side. The two nodes of each source protocol role are mapped to the

| $\tau_r$ | 1 | 2 |
|---|---|---|
| QAf | `AskA` | `BeAffirmed` |
| AnAf | `BeAskedA` | `Affirm` |
| QNg | `AskN` | `BeDenied` |
| AnNg | `BeAskedN` | `Deny` |

| $\tau_p$ | $A$ | $Q$ | $Y$ | $N$ |
|---|---|---|---|---|
| QAf | `Peer` | `Query` | `Yea` | `Nay` |
| AnAf | `Self` | `Query` | `Yea` | `Nay` |
| QNg | `Peer` | `Query` | `Yea` | `Nay` |
| AnNg | `Self` | `Query` | `Yea` | `Nay` |

Table 5: $\tau_r$ and $\tau_p$ for $\mathcal{GL}(\mathsf{YN})$

| $\tau_r$ | 1 | 2 | 3 |
|---|---|---|---|
| QAf | `GetTktA` | `AskA` | `BeAffirmed` |
| AnAf | `GiveTktA` | `BeAskedA` | `Affirm` |
| QNg | `GetTktN` | `AskN` | `BeDenied` |
| AnNg | `GiveTktN` | `BeAskedN` | `Deny` |

Table 6: $\tau_r$ for $\mathcal{GL}(\mathsf{YN}^+)$

second and third node of the $\mathsf{YN}^+$ roles. There is a transformation from HD directly to $\mathsf{YN}^+$, in which the initiator's first node is mapped to $A$'s first node, which sends the ticket, and the initiator's second node goes to $A$'s second node, with maps from the responder to $B$'s role.

We may also view $\mathsf{YN}^+$ in two ways as the target of a transformation from NS or NSL. We can map the NS initiator to the affirmative answer role or to the negative answer role. We respectively map the NS responder strand to the question role receiving the affirmative answer, or to the question role receiving the negative answer. If we take these transformations as having source NSL, they are certainly not sound, because $\mathsf{YN}^+$ certainly discloses $B$'s nonces, which NSL does not.

The language $\mathcal{GL}(\mathsf{YN}^+)$ is much like $\mathcal{GL}(\mathsf{YN})$, except that we shift the role predicates to make room for the new first node (see Table 6). We also add a new parameter predicate $\mathtt{Tkt}(n, r)$.

## 7.2 Transformations and Homomorphisms

In our approach, homomorphisms between fragments (and especially skeletons) are fundamental. Hence, we introduce a definition of transformation that is designed just to respect homomorphisms. Within this broad class of roughly reasonable operations, we will later seek a separate condition that ensures that security goals are preserved.

**Definition 7.1** *A substitution $\gamma$ is* suitable *for $\rho_1, \rho_2$ iff for some set $X$ such that* $\mathsf{Params}(\rho_1) \subseteq X$*, $\gamma$ is a bijection between $X$ and* $\mathsf{Params}(\rho_2)$*.*

So, $\gamma$ is injective going forward from parameters of $\rho_1$, and on a set of other values that it maps to the remaining parameters of $\rho_2$. When $\gamma$ is suitable, we can apply its inverse to $\rho_2$; if e.g. $\alpha(\rho_1)$ is an instance of $\rho_1$, then $\alpha(\gamma^{-1}(\rho_2))$ is a corresponding instance of $\rho_2$.

**Definition 7.2 (Transformation)** *Suppose $T$ maps each role $\rho_1 \in \Pi_1$ to a triple $\rho_2, g, \gamma$, where $\rho_2 \in \Pi_2$, $g \colon \mathbb{N}^+ \to \mathbb{N}^+$, and $\gamma$ is a substitution suitable for $\rho_1, \rho_2$. $T$ is a* protocol transformation *iff:*

1. *$g$ is order-preserving and $g(\mathsf{length}(\rho_1)) \leq \mathsf{length}(\rho_2)$;*
2. *$\rho_1 \downarrow i$ is a transmission (or resp. reception) node iff $\rho_2 \downarrow g(i)$ is;*

3. *For all parameters $x$ and integers $i$, there exists a $j \leq g(i)$ such that:*
   (a) *if $x \sqsubseteq \mathsf{msg}(\rho_1 \downarrow i)$, then $\gamma(x) \sqsubseteq \mathsf{msg}(\rho_2 \downarrow j)$; and*
   (b) *if $x$ originates on $\rho_1 \downarrow i$, then $\gamma(x)$ originates on $\rho_2 \downarrow j$;*
4. *Let $\rho_1, \sigma_1 \in \Pi_1$, let $T(\sigma_1) = \sigma_2, h, \delta$, and let $\alpha, \beta$ be substitutions. If, for every $j$ up to $i$, $\mathsf{dmsg}(\alpha(\rho_1) \downarrow j) = \mathsf{dmsg}(\beta(\sigma_1) \downarrow j)$, then:*
   (a) *$g(j) = h(j)$ for all $j \leq i$;*
   (b) *There exist $\alpha', \beta'$ that agree with $\alpha, \beta$ on the parameters of $\rho_1, \sigma_1$ respectively, where for all $j$ up to $h(i)$,*

$$\mathsf{dmsg}(\alpha'(\gamma^{-1}(\rho_2)) \downarrow j) = \mathsf{dmsg}(\beta'(\delta^{-1}(\sigma_2)) \downarrow j).$$

*When $T$ is a protocol transformation from $\Pi_1$ to $\Pi_2$, we write $T \colon \Pi_1 \to \Pi_2$.*

The first three clauses say that $T$ should preserve order, direction (send vs. receive), and where parameters are ingredients or originate. The last clause says that when the same strand can be viewed (up to height $i$) as an instance of either $\rho_1$ or $\sigma_1$, then the transformation handles it the same no matter which way we view it.

Each skeleton $\mathbb{A}$ contains nodes from a finite number of regular strands $s$. If it contains any nodes from $s$, it contains exactly the nodes $s \downarrow j$, where $1 \leq j \leq i$, for some $i$ which we call the *height* of $s$ in $\mathbb{A}$. Moreover, each regular strand $s$ is of the form $\alpha(\rho)$ for at least one role $\rho$ and substitution $\alpha$. Thus, we can always represent the nodes of a skeleton as a set of triples $\rho, \alpha, i$, each of which represents $\{(\alpha(\rho) \downarrow j) \colon 1 \leq j \leq i\}$; we avoid representing the nodes of any strand repeatedly. If a set $S$ of triples yields $\mathsf{nodes}(\mathbb{A})$, we will call $S$ a *role-substitution representation* of $\mathsf{nodes}(\mathbb{A})$.

Notice that if $\alpha, \alpha'$ differ only on parameters that do not appear in $\mathbb{A}$, then replacing $\rho, \alpha, i$ by $\rho, \alpha', i$ in a role-substitution representation of $\mathsf{nodes}(\mathbb{A})$ yields another role-substitution representation of $\mathsf{nodes}(\mathbb{A})$.

**Definition 7.3** *Assume $T \colon \Pi_1 \to \Pi_2$. When $T(\rho_1) = \rho_2, g, \gamma$, define*

$$\mathsf{lift}_T(\alpha(\rho_1) \downarrow j) = \alpha(\gamma^{-1}(\rho_2)) \downarrow g(j).$$

*That is, we first apply the inverse of $\gamma$ to find how to apply $\alpha$ to $\rho_2$; we then take the $g(j)^{\text{th}}$ node of the resulting strand.*

*Hence, if $S$ is a set of triples $\rho, \alpha, i$, define $\mathsf{lift}_T(S)$ to be the set of triples:*

$$\{(\rho_2,\ \alpha \circ \gamma^{-1},\ g(i)) \colon (\rho_1, \alpha, i) \in S \text{ and } T(\rho_1) = \rho_2, g, \gamma\}.$$

*Let $\mathbb{A}$ be a $\Pi_1$-skeleton, and let $\mathbb{C}$ be a $\Pi_2$-skeleton. $\mathbb{C}$ is a $T$-lifting of $\mathbb{A}$, written $\mathbb{C} \in \mathsf{lifts}_T(\mathbb{A})$, if:*

1. *There is a role-substitution representation $S$ of $\mathsf{nodes}(\mathbb{A})$ such that $\mathsf{lift}_T(S)$ is a role-substitution representation of $\mathsf{nodes}(\mathbb{B})$;*

2. *For all* $m, n \in \mathsf{nodes}(\mathbb{A})$, $m \preceq_{\mathbb{A}} n$ *implies* $\mathsf{lift}_T(m) \preceq_{\mathbb{B}} \mathsf{lift}_T(n)$;

3. $\mathsf{unique}_{\mathbb{A}} \subseteq \mathsf{unique}_{\mathbb{B}}$;

4. $\mathsf{non}_{\mathbb{A}} \subseteq \mathsf{non}_{\mathbb{B}}$.

We concentrate on skeletons in this section, because, while the lifting we have just defined has a canonical effect on skeletons, it does not determine how we should transform adversary strands. Those can be treated only using a far more syntactic approach to message translation. It is far from clear that protocol transformations as we have defined them always extend in a canonical way to adversary strands. Thus, we avoid fragments that are not skeletons.

Below, we write $\mathsf{strands}(\mathbb{A})$ for the set of strands $s$ with $\mathbb{A}$-height $i \geq 1$, i.e. the set of strands that have nodes in $\mathbb{A}$.

**Lemma 7.4** *Let* $\mathbb{A}$ *be a* $\Pi_1$*-skeleton, and* $T \colon \Pi_1 \to \Pi_2$.

1. *There are* $\leq_{ni}$*-minimal members* $\mathbb{C}_1$ *of* $\mathsf{lifts}_T(\mathbb{A})$.

2. *Suppose that*

$$\begin{aligned} s_1 &= \alpha(\rho_1) \in \mathsf{strands}(\mathbb{A}) \text{ and} \\ s_2 &= \beta(\sigma_1) \in \mathsf{strands}(\mathbb{A}), \text{ where} \\ \mathsf{lift}_T(s_1) &= \alpha(\gamma^{-1}(\rho_2)) \in \mathsf{strands}(\mathbb{C}_1) \text{ and} \\ \mathsf{lift}_T(s_2) &= \beta(\delta^{-1}(\sigma_2)) \in \mathsf{strands}(\mathbb{C}_1) \end{aligned}$$

   *Let* $a$ *be a parameter of* $\rho_2$*, and* $b$ *be a parameter of* $\sigma_2$ *that are not in the images of the parameters of* $\rho_1$ *under* $\gamma$ *and* $\sigma_1$ *under* $\delta$*. Then:*

   (a) $\alpha(a)$ *and* $\beta(b)$ *are both parameters;*

   (b) $\alpha(a) = \beta(b)$ *implies* $\rho_2$ *and* $\sigma_2$ *are the same, and* $a$ *and* $b$ *are also same.*

*Proof.* **1.** For each strand $s \in \mathsf{strands}(\mathbb{A})$, letting $s = \alpha(\rho_1)$, select an instance $\lambda(s) = \beta_s(\rho_2)$, of height $g(i)$, where the instantiation $\beta_s$ associates not-yet-used parameters with each parameter of $\rho_2$. Let the $\Pi_2$-skeleton $\mathbb{C}_0$ be the resulting skeleton, where $n_0 \preceq_{\mathbb{C}_0} n_1$ implies $n_0 \Rightarrow^+ n_1$. We take $\mathsf{unique}_{\mathbb{C}_0}, \mathsf{non}_{\mathbb{C}_0}$ to be the minimal possible sets, i.e. $\mathsf{unique}_{\mathbb{C}_0}$ is the union over the strands $s = \alpha(\rho_1)$ in $\mathsf{strands}(\mathbb{A})$ of $\beta_s(\mathsf{role\_unique}(\rho_2))$. Likewise, $\mathsf{non}_{\mathbb{C}_0}$ is the union over the strands $s = \alpha(\rho_1)$ in $\mathsf{strands}(\mathbb{A})$ of $\beta_s(\mathsf{role\_non}(\rho_2))$.

There is a node-injective homomorphism $H_{\mathbb{C}} \colon \mathbb{C}_0 \hookrightarrow_{ni} \mathbb{C}$ for each $\mathbb{C} \in \mathsf{lifts}_T(\mathbb{A})$.

Using these homomorphisms, apply Lemma 5.11, clause 3 repeatedly, once for each pair of nodes $\lambda(n), \lambda(m)$ such that $n \preceq_{\mathbb{A}} m$. Use Lemma 5.11, clause 1 repeatedly, once for each $\beta_s \circ \gamma^{-1}(a)$ and $\alpha(a)$ where (i) $s \in \mathsf{strands}(\mathbb{A})$; (ii) $s = \alpha(\rho_1)$; (iii) $a$ is a parameter to $\rho_1$; and (iv) $T(\rho_1) = \rho_2, g, \gamma$.

At every stage, the resulting skeleton has a node-injective homomorphism to every skeleton in $\mathsf{lifts}_T(\mathbb{A})$. When we are finished, the resulting skeleton $\mathbb{C}_1$ is in $\mathsf{lifts}_T(\mathbb{A})$. Thus, it is $\leq_{ni}$-minimal within $\mathsf{lifts}_T(\mathbb{A})$.

**2.** The statement is true for $\mathbb{C}_0$, and it remains true under each step of applying Lemma 5.11, clause 1, which affect only the parameters $\gamma(c)$ where $c$ is a parameter of $\rho_1$. □

Since $\mathbb{A} \leq_{ni} \mathbb{B} \leq_{ni} \mathbb{A}$ implies that $\mathbb{A}, \mathbb{B}$ are isomorphic, we may define $T(\mathbb{A})$:

**Definition 7.5** *$T(\mathbb{A})$ is the $\leq_{ni}$-minimal member of $\mathsf{lifts}_T(\mathbb{A})$, which is unique to within isomorphism.*

The following theorem justifies our definition of transformations.

**Theorem 7.6** *Let $T\colon \Pi_1 \to \Pi_2$.*

1. *If $H\colon \mathbb{A} \to \mathbb{B}$ is a $\Pi_1$-homomorphism, there is a $\Pi_2$-homomorphism $G\colon T(\mathbb{A}) \to T(\mathbb{B})$ such that, for every $n \in \mathsf{nodes}(\mathbb{A})$,*

$$\mathsf{lift}_T(H(n)) = G(\mathsf{lift}_T(n)).$$

   *Moreover, if $G$ and $G'$ both satisfy this property, then they differ by an isomorphism $I$, i.e. $G' = I \circ G$. $G$ is node-injective iff $H$ is.*

2. *If instead $G\colon T(\mathbb{A}) \to T(\mathbb{B})$, then there is an $H\colon \mathbb{A} \to \mathbb{B}$ such that $G = F(H)$. $H$ is unique to within isomorphism.*

*Proof.* **1.** Let $H = f, \beta$. Viewing the image of $\mathsf{strands}(\mathbb{A})$ under $f$, each of those strands is of the form $\beta(\alpha(\rho_1))$, for some $\rho_1$ and $\alpha$, where the strand in $\mathsf{strands}(\mathbb{A})$ is of the form $\alpha(\rho_1)$. Thus, each strand in $T(\mathbb{A})$ is of the form $\alpha(\gamma(\rho_2)^{-1})$, where $T(\rho_1) = \rho_2, g, \gamma$. Thus, we can use homomorphism $G$ with node function and substitution

$$\begin{array}{ccc} T(\mathbb{A}) & \xrightarrow{G} & T(\mathbb{B}) \\ {\scriptstyle \mathsf{lift}_T}\uparrow & & \uparrow{\scriptstyle \mathsf{lift}_T} \\ \mathbb{A} & \xrightarrow{H} & \mathbb{B} \end{array}$$

$$\mathsf{lift}_T \circ f \circ {\mathsf{lift}_T}^{-1} \quad \text{and} \quad \beta.$$

This construction is canonical, and is node-injective if $f$ is.

**2.** If $G = f, \beta$, then

$$\mathsf{lift}_T^{-1} \circ f \circ \mathsf{lift}_T \quad \text{and} \quad \beta$$

work for $H$. □

## 7.3 Preservation via PSLTSs

Let $\Phi \in \mathcal{GL}(\Pi_1)$ be role-specific, and let $G_\Phi$ be the set of goals of the form $\forall \vec{x}\,.\,(\Phi \longrightarrow \Psi)$, as $\Psi$ varies over positive existential formulas.

To show that the goals $G_\Phi$ are preserved under a transformation $T\colon \Pi_1 \to \Pi_2$, we would like to exhibit a PSLTS for $\Pi_1$ that contains $\mathsf{cfrag}(\Phi)$, and a PSLTS for $\Pi_2$ that contains $T(\mathsf{cfrag}(\Phi))$. If these PSLTSs match up, then all the goals $G_\Phi$ will be preserved. In this subsection, we will define what "match up" means here, and we will prove a structural theorem that relates the how the two PSLTSs

reach realized skeletons. It generalizes Thm. 6.4, at least for skeletons; Thm. 6.4 covers the case of the identity transformation $\mathsf{Id}\colon \Pi_1 \to \Pi_1$.

One degree of freedom concerns the labels. We care little about their structure. We assume simply that each PSLTS comes with its labels $\Lambda_1, \Lambda_2$, each containing the distinguished label dead. We allow a map between the labels as a relabeling function if it respects the dead. We also need the relevant notions of progress and simulation:

**Definition 7.7** *Let $\Delta\colon \Lambda_1 \to \Lambda_2$ be a function between the $\Lambda_i$, and let $T\colon \Pi_1 \to \Pi_2$ be a protocol transformation. Let $\rightsquigarrow_1$ be a* PSLTS *on $\Pi_1$ using labels $\Lambda_1$; let $\rightsquigarrow_2$ be a* PSLTS *on $\Pi_2$ using labels $\Lambda_2$.*

1. *$\Delta$ is a* relabeling function *iff $\Delta^{-1}(\{\mathsf{dead}\}) = \{\mathsf{dead}\}$.*

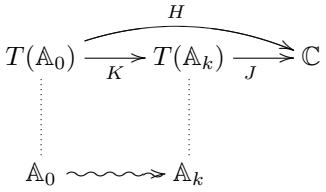
*Let $\Delta$ be a relabeling function.*

2. *$T, \Delta$* preserve progress *for $\rightsquigarrow_1$ and $\rightsquigarrow_2$ iff, for every $\ell \in \Lambda$, $\mathbb{A} \overset{\ell}{\rightsquigarrow_1} \cdot$ implies $T(\mathbb{A}) \overset{\Delta(\ell)}{\rightsquigarrow_2} \cdot$.*

3. *$\rightsquigarrow_1$* simulates *$\rightsquigarrow_2$ under $T, \Delta$ iff: $T(\mathbb{A}) \overset{\ell'}{\rightsquigarrow_2} \mathbb{B}'$ and $\ell' = \Delta(\ell)$ entails $\exists \mathbb{B}$ s.t. $\mathbb{B}' = T(\mathbb{B})$ and $\mathbb{A} \overset{\ell}{\rightsquigarrow_1} \mathbb{B}$.*

**Theorem 7.8 (Preservation)** *Let $\Delta\colon \Lambda_1 \to \Lambda_2$ be a relabeling function and let $T\colon \Pi_1 \to \Pi_2$ be a protocol transformation. Let $\rightsquigarrow_1$ and $\rightsquigarrow_2$ be* PSLTS*s for $\Pi_1$ and $\Pi_2$ resp., with $\mathbb{A}_0 \in S(\rightsquigarrow_1)$ and $T(\mathbb{A}_0) \in S(\rightsquigarrow_2)$. Suppose that:*

1. *$T, \Delta$ preserve progress for $\rightsquigarrow_1$, $\rightsquigarrow_2$;*
2. *$\rightsquigarrow_1$ simulates $\rightsquigarrow_2$ under $T, \Delta$.*




*For every $\Pi_2$-realized $\mathbb{C}$, if $H\colon T(\mathbb{A}_0) \to \mathbb{C}$, there is a $\Pi_1$-realized $\mathbb{A}_k \in \mathsf{Skel}(\Pi_1)$ such that $\mathbb{A}_0 \rightsquigarrow_1^* \mathbb{A}_k$, and $H$ factors through $T(\mathbb{A}_k)$.*

*Proof.* Let $\Sigma$ be the set of $\rightsquigarrow_1$ paths $\sigma$ of the form:

$$\mathbb{A}_0 \overset{\ell_1}{\rightsquigarrow_1} \cdots \overset{\ell_i}{\rightsquigarrow_1} \mathbb{A}_i \overset{\ell_{i+1}}{\rightsquigarrow_1} \cdots \overset{\ell_j}{\rightsquigarrow_1} \mathbb{A}_j$$

starting at $\mathbb{A}_0$. Let $\Sigma_H$ be the set of $\sigma \in \Sigma$ such that, for some $J\colon T(\mathbb{A}_j) \to \mathbb{C}$, we have $H = J \circ H_\sigma$. These $\sigma$ are the ones that never diverge from $H$. $\Sigma_H$ is non-empty since the empty path is in it.

By Defn. 6.3, Clause 3b, there are maximal members in $\Sigma_H$, i.e. $\sigma_1$ such that no extension $\sigma_1^\frown \ell$ is in $\Sigma_H$.

So let $\sigma_1$ be maximal in $\Sigma_H$, where $\mathbb{A}_0 \overset{\sigma_1}{\rightsquigarrow_1^*} \mathbb{A}_j$. By construction, $T(H_{\sigma_1})\colon T(\mathbb{A}_0) \to T(\mathbb{A}_j)$, and $H = J \circ H_{\sigma_1}$. So we need only check that $\mathbb{A}_j$ is realized.

However, if $\mathbb{A}_j$ is not realized, then by Defn. 6.3, Clause 1, there is a label $\ell$ and skeleton $\mathbb{A}_{j+1}$ such that $\mathbb{A}_j \overset{\ell}{\rightsquigarrow}_1 \mathbb{A}_{j+1}$. By progress (assumption 1), $T(\mathbb{A}_j) \overset{\Delta(\ell)}{\rightsquigarrow}_2 T(\mathbb{A}_{j+1})$, so $T(\mathbb{A}_j)$ is unrealized.

Moreover, $\ell \neq \mathsf{dead}$: If $T(\mathbb{A}_j) \overset{\mathsf{dead}}{\rightsquigarrow}_2 T(\mathbb{A}_{j+1})$, then Defn. 6.3, Clause 2 contradicts the assumption that $J\colon T(\mathbb{A}_j) \rightarrowtail \mathbb{C}$.

Using Clause 3a, there is in fact a $\mathcal{E}'$ such that $T(\mathbb{A}_j) \overset{\Delta(\ell)}{\rightsquigarrow}_2 \mathcal{E}'$ and $J = T(\mathbb{A}_j) \overset{H_\ell}{\rightarrowtail} \mathcal{E}' \overset{K}{\rightarrowtail} \mathcal{C}$. That is, $H_{\Delta(\ell)}\colon T(\mathbb{A}_j) \rightarrowtail \mathcal{E}'$ is the member of the $\Delta(\ell)$ cohort compatible with $J$. So by simulation (assumption 2), it follows that there is a $\mathbb{B}$ such that $T(\mathbb{B}) = \mathcal{E}'$ and $\mathbb{A}_j \overset{\ell}{\rightsquigarrow}_1 \mathbb{B}$. So in fact

$$\mathbb{A}_0 \overset{\ell_1}{\rightsquigarrow}_1 \cdots \overset{\ell_i}{\rightsquigarrow}_1 \mathbb{A}_i \overset{\ell_{i+1}}{\rightsquigarrow}_1 \cdots \overset{\ell_j}{\rightsquigarrow}_1 \mathbb{A}_j \overset{\ell}{\rightsquigarrow}_1 \mathbb{B}$$

is also in $\Sigma_H$, contradicting the maximality of $\sigma$. Hence, $\mathbb{A}_j$ is realized. □

The proof of Thm. 6.4 is essentially this proof, letting $T, \Delta$ be the identity.

## 7.4 Preserving Goal Formulas

Each $T\colon \Pi_1 \to \Pi_2$ determines a translation from role specific goal formulas $\phi$ of $\mathcal{GL}(\Pi_1)$ into $\mathcal{GL}(\Pi_2)$. We define this translation in the simplest possible way, going through $\phi$ one atomic formula at a time and replacing some of the predicate symbols. We use the tables $\tau_r$ and $\tau_p$ for the two languages to determine how to do this replacement.

By the definition of *role specific*, Defn. 5.8, if a variable $v$ occurs in $\phi$, then it is either a node variable or a message variable. A node variable appears in some role position predicate $R(v)$, and a message variable occurs as the second argument to a parameter predicate $P(n, v)$.

If $R$ is a role position predicate, then we say that the *role position location of* $R$ is $(\rho, i)$ if in the table $\tau_r$ for $\mathcal{GL}(\Pi)$, the entry for $(\rho, i)$ is $R$. Given $\phi$ and a node variable $n$ in $\phi$, we will say that its *role position location in* $\phi$ is $(\rho, i)$ if the leftmost role position predicate in $\phi$ containing $n$ is $R(n)$, and the role position location of $R$ is $(\rho, i)$.

If $P(n, t)$ is an atomic formula where $P$ is a parameter predicate and $n$ is a variable, then we say that $P(n, t)$ has *role parameter* $(\rho, a)$ in $\phi$ if:

- $n$ has role position location $(\rho, i)$ in $\phi$, for some $i$; and
- in the table $\tau_p$ for $\mathcal{GL}(\Pi)$, the entry for $(\rho, a)$ is $P$.

If $P(t', t)$ has a non-variable $t'$ in the first position, then $t'$ is the result of a key function, and the formula $P(t', t)$ will be false in all interpretations.

**Definition 7.9** *Let $T\colon \Pi_1 \to \Pi_2$. The $T$-translation of a role specific formula $\phi$ is the result of replacing each predicate symbol $R$ within it according to the following rules:*

- *If $R$ belongs to the protocol-independent vocabulary $=$, $\mathtt{Preceq}$, $\mathtt{Coll}$, $\mathtt{Unq}$, $\mathtt{UnqAt}(n, v)$, $\mathtt{Non}(v)$, then it is unchanged.*
- *Suppose $R$ is a role position predicate, and the role position location of $R$ is $(\rho_1, i)$, and let $T(\rho_1) = \rho_2, g, \gamma$.*

  *Replace $R$ with $R'$, which is the entry in $\tau_r$ for $\mathcal{GL}(\Pi_2)$ for $\rho_2$ and $g(i)$.*
- *Suppose that $R$ is a parameter predicate, and appears in the form $R(n, t)$. Let $R(n, t)$ have parameter $(\rho, a)$ in $\phi$, and let $T(\rho_1) = \rho_2, g, \gamma$.*

  *Replace $R$ with $R'$, which is the entry in $\tau_p$ for $\mathcal{GL}(\Pi_2)$ for $\rho_2$ and $\gamma(a)$.*
- *Suppose that $R$ is a parameter predicate, and appears in the form $R(t', t)$, where $t'$ is not a variable. Then it will be false.*

  *Replace $R$ with any $R'$ where $R'$ is a parameter predicate in $\mathcal{GL}(\Pi_2)$.*

*We write $T(\phi)$ for the result of this process.*

*If $\forall \vec{x} \,.\, (\phi \longrightarrow \bigvee_i \exists \vec{y_i} \,.\, \psi_i)$ is a goal formula $\Gamma$ with role specific $\phi$, we write $T(\Gamma)$ for the formula $\forall \vec{x} \,.\, (T(\phi) \longrightarrow \bigvee_i \exists \vec{y_i} \,.\, T(\psi_i))$.*

We may tacitly repeat the hypothesis $\phi$ when it would be convenient: so $T(\Gamma)$ means $\forall \vec{x} \,.\, (T(\phi) \longrightarrow \bigvee_i \exists \vec{y_i} \,.\, T(\phi \wedge \psi_i))$, when the $\psi_i$ are not role specific.

$T(\phi)$ is role specific when $\phi$ is, and $T(\Gamma)$ is a goal formula when $\Gamma$ is.

**Definition 7.10** *Let $T \colon \Pi_1 \to \Pi_2$ and $\mathbb{A} \in \mathsf{Skel}(\Pi_1)$, and let $\eta$ be a variable assignment $\eta \colon \mathsf{Var} \to \mathsf{nodes}(\mathbb{A}) \cup \mathsf{ALG}$. Let $\mathsf{lift}_T$ lift $\mathsf{nodes}(\mathbb{A})$ to $\mathsf{nodes}(T(\mathbb{A}))$.*

*The* extension of *$\eta$ for $\mathbb{A}, T$ is the function which, for a variable $x$, returns $\eta(x)$ if the latter is in $\mathsf{ALG}$, and returns $\mathsf{lift}_T(\eta(x))$ if $\eta(x) \in \mathsf{nodes}(\mathbb{A})$.*

When $\mathbb{A}, T$ are clear, we write $\overline{\eta}$ for the extension of $\eta$ for $\mathbb{A}, T$.

**Lemma 7.11** *Let $T \colon \Pi_1 \to \Pi_2$, $\mathbb{A} \in \mathsf{Skel}(\Pi_1)$, $\mathbb{B} \in \mathsf{Skel}(\Pi_2)$, and let $\phi \in \mathcal{GL}(\Pi_1)$ be a satisfiable, role-specific conjunction of atomic formulas.*

1. *$\mathsf{cfrag}(T(\phi))$ exists.*
2. *If $\mathbb{A}, \eta \models \phi$, then $T(\mathbb{A}), \overline{\eta} \models T(\phi)$.*
3. *If $\mathbb{B}, \theta \models T(\phi)$ and $\mathsf{cfrag}_\eta(\phi) = \mathbb{A}$, then there exists a $J \colon T(\mathbb{A}) \to \mathbb{B}$ such that $\theta \sim_\phi J \circ \overline{\eta}$.*
4. *$T(\mathsf{cfrag}(\phi)) = \mathsf{cfrag}(T(\phi))$.*

*Proof.* **1.** As observed, $T(\phi)$ is role specific, so Thm. 5.12 applies.

**2.** By induction on the structure of the conjunction $\phi$. Essentially, one checks that Defn. 7.9 matches Defn. 7.3.

**3.** Define $J = [f, \alpha]$: For $f$, let $f(\overline{\eta}(x)) = \theta(x)$, observing that every strand in $\mathbb{A}$ has a node in range$(\eta)$, whence every strand in $T(\mathbb{A})$ has a node in range$(\overline{\eta})$. For $\alpha$, let $a$ be any parameter of $\mathbb{A}$. Hence, there is some $s = \beta(\rho_1)$ and

$b \in \mathsf{Params}(\rho_1)$ such that $a = \beta(b)$. Letting $f(s) = \delta(\rho_2)$, set $\alpha(\gamma(b)) = \delta(\gamma(b))$, where as usual $T(\rho_1) = \rho_2, g, \gamma$.

**4.** By the previous clause, we have a $J \colon T(\mathsf{cfrag}(\phi)) \to \mathsf{cfrag}(T(\phi))$.

By clause 2, $T(\mathsf{cfrag}(\phi)), \overline{\eta} \models T(\phi)$. Thus, Def. 5.5 entails that there is a $K \colon \mathsf{cfrag}(T(\phi)) \to T(\mathsf{cfrag}(\phi))$. Hence, by the uniqueness in Def. 5.5, $J \circ K = \mathsf{Id}$. Hence, $T(\mathsf{cfrag}(\phi))$ and $\mathsf{cfrag}(T(\phi))$ are isomorphic. □

We now turn to our last main theorem.

**Theorem 7.12 (Goal Preservation)** *Let $\phi$ be a role specific conjunction of atomic formulas. Let $\Delta \colon \Lambda_1 \to \Lambda_2$ be a relabeling function and let $T \colon \Pi_1 \to \Pi_2$ be a protocol transformation. Let $\rightsquigarrow_1$ and $\rightsquigarrow_2$ be* PSLTS*s for $\Pi_1$ and $\Pi_2$ resp., with $\mathsf{cfrag}(\phi) \in S(\rightsquigarrow_1)$ and $T(\mathsf{cfrag}(\phi)) \in S(\rightsquigarrow_2)$. Suppose*

1. *$T, \Delta$ preserve progress for $\rightsquigarrow_1$, $\rightsquigarrow_2$;*
2. *$\rightsquigarrow_1$ simulates $\rightsquigarrow_2$ under $T, \Delta$.*

*Then for every security goal $\Gamma = \forall \overline{x} \,.\, \phi \longrightarrow \bigvee_i \exists \overline{y_i} \,.\, \psi_i$, if $\Pi_1$ achieves $\Gamma$, then $\Pi_2$ achieves $T(\Gamma)$.*

*Proof.* $T(\Gamma) = \forall \overline{x} \,.\, T(\phi) \longrightarrow \bigvee_i \exists \overline{y_i} \,.\, T(\psi_i)$.

Suppose that $\mathbb{C}$ is any realized $\Pi_2$-skeleton, and $\theta$ is a variable assignment into $\mathsf{nodes}(\mathbb{C}) \cup \mathsf{ALG}$. If $\mathbb{C}, \theta \not\models T(\phi)$, then $\mathbb{C}, \theta \models T(\phi) \longrightarrow \bigvee_i \exists \overline{y_i} \,.\, T(\psi_i)$.

So suppose $\mathbb{C}, \theta \models T(\phi)$. By Defn. 5.5, there is an $H \colon \mathsf{cfrag}_{\overline{\eta}}(T(\phi)) \to \mathbb{C}$, and $\theta \sim_\phi H \circ \overline{\eta}$. Moreover,

$$\mathsf{cfrag}_{\overline{\eta}}(T(\phi)) = T(\mathsf{cfrag}_{\eta}(\phi)),$$

by Lemma 7.11, Clause 4. So we may apply Thm. 7.8: There is a $\Pi_1$-realized $\mathbb{B}$ such that $\mathsf{cfrag}_{\eta}(\phi) \overset{\sigma}{\rightsquigarrow}_1^* \mathbb{B}$, and, for $K = T(H_\sigma)$,

$$T(\mathsf{cfrag}_{\eta}(\phi)) \overset{K}{\to} T(\mathbb{B}) \overset{J}{\to} \mathbb{C}.$$

If $\Pi_1$ achieves $\Gamma$, then for some $i$, $\psi_i$ is satisfied in $\mathbb{B}$, i.e. $\mathbb{B}, \zeta \models \psi_i$ where $\zeta \sim_\phi (H_\sigma \circ \eta)$. Hence, $T(\mathbb{B}), \overline{\zeta} \models T(\psi_i)$. Moreover, $\overline{\zeta} \sim_{T(\phi)} (K \circ \overline{\eta})$. Since homomorphisms preserve satisfaction for positive existential formulas (Lemma 4.3), $\mathbb{C}, J \circ \overline{\zeta} \models T(\psi_i)$, where $J \circ \overline{\zeta} \sim_{T(\phi)} (H \circ \overline{\eta})$. □

# 8 Conclusion

The safe protocol transformation problem is not new. As an idea for protocol design, it goes back at least to Bird et al. [7]. In a key special case, "protocol composition," it dates from the 1990s [41, e.g.]. In the protocol composition case, roles of $\Pi_1$ also appear unchanged as roles of $\Pi_2$. Since $\Pi_2$ may also have additional roles not in the image of $\Pi_2$, composition is thus effectively the case

in which $\Pi_1 \subseteq \Pi_2$. While there has been an extensive literature devoted to this special case, the more general type of transformation discussed here has seen very little progress. Our view is that the effects of a syntactic change in message structure on protocol behavior are very hard to predict (given an active adversary model). This has made it hard to reason about the full notion of transformation, as opposed to the special case of composition. We have introduced the PSLTS as a representation of the protocol analysis problem to tame this complexity.

Focusing then on protocol composition, it has been very successfully treated in a *cryptographic* model. A strong form of composition is reactive simulatability [52, 5] or universal composability [12]; weaker forms may still be cryptographically justified [21].

In the *symbolic* model, we previously provided a widely applicable and practically useful criterion [38, 32]. Cortier et al.'s criterion is in some ways broader but in other ways narrower than ours [14]; cf. [3]. Our [33] covers the union of [38, 14, 3]. From one point of view, the contribution of the present paper is to generalize [33] beyond the composition case.

The Protocol Composition Logic PCL considers refinements that preserve security goals [20, Thms. 4.4, 4.8]. A specific proof of a goal formula relies on particular invariants. If a protocol refinement introduces no actions falsifying these invariants, it preserves the security goal. Although PCL was designed to support richer forms of transformation, the existing results are essentially confined to the composition case. [20]'s "parallel" and "sequential" composition amounts to $\Pi_1 \subseteq \Pi_2$. Datta et al.'s "protocol refinement using templates" [19] suggested many of our examples.

By contrast with Distributed Temporal Logic [11], $\mathcal{GL}(\Pi)$ is intended to be less expressive about the forms of messages. We wanted to focus only on what is retained under transformation, which concerns the role parameters rather than the forms of the messages. Nevertheless, our logic, unlike DTL, being a quantified logic, satisfaction is undecidable.

Lowe and Auty [45] refine protocols to concrete messages starting from formulas in a Hoare-like logic that represent the effect of messages. Maffei et al. [4] express the effects of messages by abstract tags, and provide constraints on instantiating the tags by concrete messages.

"Protocol compilers" transform their input automatically. Some start with a crypto-free protocol, and transform it into a protocol meeting security goals [15, 6]. Others transform a protocol secure in a weak adversary model into protocols satisfying those goals with multi-session, active adversary [40].

**Future work.** We leave a major gap: What syntactic property of $F\colon \Pi_1 \to \Pi_2$ ensures that $F$ preserves security goals? A clue comes from the "disjoint encryption" property [38, 33], cf. [45, 14]. Consider a map $E$ from all encrypted units used by $\Pi_1$ to a subset of the encrypted units of $\Pi_2$. $\Pi_2$ should create an encryption $\alpha(E(e))$ on node $n$ only if $n = F(n_0)$ and $n_0$ creates $\alpha(e)$ in $\Pi_1$. Likewise, $\Pi_2$ should remove an ingredient from $\alpha(E(e))$ only on a node

$n = F(n_0)$ where $n_0$ removes an ingredient from $\alpha(e)$ in $\Pi_1$.

Tool support is also required. CPSA generates some PSLTS transition relations. We then construct others, and the simulations, by hand. A variant of CPSA that would explore two protocols in tandem would be of great interest.

**Acknowledgments.** Thanks to Dan Dougherty, Dusko Pavlovic, John Ramsdell, Paul Rowe, and Javier Thayer. Early versions of some of this material were presented at FCS-ARPSA-WITS in 2008 and in Darmstadt in 2010.

Thanks to Siraj Sayani and Soumentra Ghosal, whose hospitality I enjoyed in Coonoor while writing a key part of this paper.

# References

[1] Martín Abadi and Bruno Blanchet. Analyzing security protocols with secrecy types and logic programs. *Journal of the ACM*, 52(1):102–146, January 2005.

[2] Roberto M. Amadio and Denis Lugiez. On the reachability problem in cryptographic protocols. In *Concur*, number 1877 in LNCS, pages 380–394, 2000.

[3] S. Andova, C.J.F. Cremers, K. Gjøsteen, S. Mauw, S.F. Mjølsnes, and S. Radomirović. Sufficient conditions for composing security protocols. *Information and Computation*, 2007.

[4] Michael Backes, Agostino Cortesi, Riccardo Focardi, and Matteo Maffei. A calculus of challenges and responses. In *FMSE '07: ACM Workshop on Formal methods in Security Engineering*, pages 51–60, New York, NY, USA, 2007. ACM.

[5] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A universally composable cryptographic library. Available at `http://eprint.iacr.org/2003/015/`, January 2003.

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

[7] R. Bird, I. Gopal, A. Herzberg, P. A. Janson, S. Kutten, R. Mulva, and M. Yung. Systematic design of a family of attack-resistant authentication protocols. *IEEE Journal on Selected Areas in Communications*, 11(5):679–693, 1993.

[8] Simon Blake-Wilson and Alfred Menezes. Authenticated Diffe-Hellman key agreement protocols. In *Selected Areas in Cryptography*, pages 630–630. Springer, 1999.

[9] Bruno Blanchet. An efficient protocol verifier based on Prolog rules. In *14th Computer Security Foundations Workshop*, pages 82–96. IEEE CS Press, June 2001.

[10] Dan Boneh. The decision Diffie-Hellman problem. *Algorithmic Number Theory*, pages 48–63, 1998.

[11] C. Caleiro, L. Vigano, and D. Basin. Relating strand spaces and distributed temporal logic for security protocol analysis. *Logic Journal of IGPL*, 13(6):637, 2005.

[12] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Report 2000/067, International Association for Cryptographic Research, October 2001. Extended Abstract appeared in proceedings of the 42nd Symposium on Foundations of Computer Science (FOCS), 2001.

[13] Edmund Clarke, Somesh Jha, and Will Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In *Proceedings, IFIP Working Conference on Programming Concepts and Methods (*PROCOMET*)*, 1998.

[14] Véronique Cortier, Jérémie Delaitre, and Stéphanie Delaune. Safely composing security protocols. In V. Arvind and Sanjiva Prasad, editors, *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, LNCS, New Delhi, India, December 2007. Springer.

[15] Véronique Cortier, Bogdan Warinschi, and Eugen Zalinescu. Synthesizing secure protocols. In *ESORICS: European Symposium On Research In Computer Security*, volume 4734 of *Lecture Notes in Computer Science*, pages 406–421. Springer, 2007.

[16] Federico Crazzolara and Glynn Winskel. Composing strand spaces. In *Proceedings, Foundations of Software Technology and Theoretical Computer Science*, number 2556 in LNCS, pages 97–108, Kanpur, December 2002. Springer Verlag.

[17] Cas J.F. Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *ACM Conference on Computer and Communications Security (CCS)*, pages 119–128, New York, NY, USA, 2008. ACM.

[18] C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Ph.D. dissertation, Eindhoven University of Technology, 2006.

[19] Anupam Datta, Ante Derek, John C. Mitchell, and Dusko Pavlovic. Abstraction and refinement in protocol derivation. In *IEEE Computer Security Foundations Workshop*. IEEE CS Press, 2004.

[20] Anupam Datta, Ante Derek, John C. Mitchell, and Dusko Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 13(3):423–482, 2005.

[21] Anupam Datta, Ante Derek, John C. Mitchell, and Bogdan Warinschi. Computationally sound compositional logic for key exchange protocols. In *Computer Security Foundations Workshop*, pages 321–334, 2006.

[22] Dorothy Denning and G. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8), August 1981.

[23] 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, 2007.

[24] Daniel J. Dougherty and Joshua D. Guttman. Symbolic protocol analysis for Diffie-Hellman. *Arxiv preprint arXiv:1202.2168*, 2012. At http://arxiv.org/abs/1202.2168v1.

[25] Nancy Durgin, Patrick Lincoln, John Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004. Initial version appeared in *Workshop on Formal Methods and Security Protocols*, 1999.

[26] Santiago Escobar, Catherine Meadows, and José Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. *Foundations of Security Analysis and Design V*, pages 1–50, 2009.

[27] Marcelo Fiore and Martín Abadi. Computing symbolic models for verifying cryptographic protocols. In *Computer Security Foundations Workshop*, June 2001.

[28] Andrew D. Gordon and Alan Jeffrey. Authenticity by typing for security protocols. *Journal of Computer Security*, 11(4):451–521, 2003.

[29] Andrew D. Gordon and Alan Jeffrey. Types and effects for asymmetric cryptographic protocols. *Journal of Computer Security*, 12(3/4):435–484, 2004.

[30] Jean Goubault-Larrecq. Towards producing formally checkable security proofs, automatically. In *Computer Security Foundations Workshop*, pages 224–238, 2008.

[31] Joshua D. Guttman. Key compromise and the authentication tests. *Electronic Notes in Theoretical Computer Science*, 47, 2001. Editor, M. Mislove. URL http://www.elsevier.nl/locate/entcs/volume47.html, 21 pages.

[32] Joshua D. Guttman. Authentication tests and disjoint encryption: a design method for security protocols. *Journal of Computer Security*, 12(3/4):409–433, 2004.

[33] Joshua D. Guttman. Cryptographic protocol composition via the authentication tests. In Luca de Alfaro, editor, *Foundations of Software Science and Computation Structures (FOSSACS)*, number 5504 in LNCS, pages 303–317. Springer, March 2009.

[34] Joshua D. Guttman. Security theorems via model theory. *EXPRESS: Expressiveness in Concurrency (EPTCS)*, 8:51, 2009. doi:10.4204/EPTCS.8.5.

[35] 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.

[36] Joshua D. Guttman. Security goals and protocol transformations. In Sebastian Mödersheim and Catuscia Palamidessi, editors, *Tosca: Theory of Security and Applications*, LNCS. Springer, March 2011.

[37] Joshua D. Guttman. Shapes: Surveying crypto protocol runs. In Veronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, Cryptology and Information Security Series. IOS Press, 2011.

[38] Joshua D. Guttman and F. Javier Thayer. Protocol independence through disjoint encryption. In *Computer Security Foundations Workshop*. IEEE CS Press, 2000.

[39] Joshua D. Guttman and F. Javier Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 283(2):333–380, June 2002. Conference version appeared in *IEEE Symposium on Security and Privacy*, May 2000.

[40] Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. *J. Cryptology*, 20(1):85–113, 2007.

[41] John Kelsey, Bruce Schneier, and David Wagner. Protocol interactions and the chosen protocol attack. In *Security Protocols Workshop*. Springer, 1998.

[42] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceeedings of* TACAS, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer Verlag, 1996.

[43] Gavin Lowe. Casper: A compiler for the analysis of security protocols. In *10th Computer Security Foundations Workshop Proceedings*, pages 18–30. IEEE CS Press, 1997.

[44] Gavin Lowe. A hierarchy of authentication specifications. In *10th Computer Security Foundations Workshop Proceedings*, pages 31–43. IEEE CS Press, 1997.

[45] Gavin Lowe and Michael Auty. A calculus for security protocol development. Technical report, Oxford University Computing Laboratory, March 2007.

[46] C. Meadows. The NRL protocol analyzer: An overview. *The Journal of Logic Programming*, 26(2):113–131, 1996.

[47] J. K. Millen, S. C. Clark, and S. B. Freedman. The Interrogator: Protocol security analysis. *IEEE Transactions on Software Engineering*, 13(2):274–288, February 1987.

[48] Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *8th ACM Conference on Computer and Communications Security (CCS '01)*, pages 166–175. ACM, 2001.

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

[50] Lawrence C. Paulson. Proving properties of security protocols by induction. In *10th IEEE Computer Security Foundations Workshop*, pages 70–83. IEEE CS Press, 1997.

[51] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 1998. Also Report 443, Cambridge University Computer Lab.

[52] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *Proceedings, Seventh ACM Conference of Communication and Computer Security*. ACM, November 2000.

[53] John D. Ramsdell. Deducing security goals from shape analysis sentences. The MITRE Corporation, April 2012. http://arxiv.org/abs/1204.0480.

[54] John D. Ramsdell and Joshua D. Guttman. CPSA: A cryptographic protocol shapes analyzer. In *Hackage*. The MITRE Corporation, 2009. http://hackage.haskell.org/package/cpsa; see esp. `doc` subdirectory.

[55] John D. Ramsdell, Joshua D. Guttman, and Paul D. Rowe. *The CPSA Specification: A Reduction System for Searching for Shapes in Cryptographic Protocols*. The MITRE Corporation, 2009. In http://hackage.haskell.org/package/cpsa source distribution, `doc` directory.

[56] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Computer Security Foundations Workshop*, pages 174–, 2001.

[57] Peter Selinger. Models for an adversary-centric protocol logic. *Electr. Notes Theor. Comput. Sci.*, 55(1), 2001.

[58] Dawn Xiaodong Song. Athena: a new efficient automated checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*. IEEE CS Press, June 1999.

[59] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.