Establishing and Preserving Protocol Security Goals [*]

Joshua D. Guttman

October 8, 2013

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 *geometric sequents*, i.e. implications $\Phi \longrightarrow \Psi$ where Φ and Ψ are built from atomic formulas without negations, implications, or universal quantifiers.

Security goals are then statements about homomorphisms, where the source is a minimal (fragmentary) model of the antecedent Φ. If every homomorphism to a non-fragmentary, complete execution factors through a model in which Ψ is satisfied, then the goal is achieved. One can validate security goals via a process of information enrichment. We call this approach *enrich-by-need* protocol analysis.

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.

Contents

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

Yet why not say what happens?
– "Epilogue," R. Lowell

1 Introduction

We focus in this paper on *enrich-by-need* protocol analysis. This is a style of symbolic analysis for security protocols in which the analysis proceeds by exploring fragments of possible executions. It aims to build a representative catalog of all the essentially different things that can happen when a protocol can execute. Since a protocol has infinitely many different executions (if we count executions in which any n separate sessions have occurred), enrich-by-need protocol analysis catalogs only the minimal, essentially different portions of large executions.

We regard all executions as being *models*, i.e. first order *structures* 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 all these structures *fragments*, and the self-contained ones *realized fragments*. Our representative catalog of executions should contain only realized fragments, and it should offer suitable substructures for all larger executions.

Enrich-by-need protocol analysis is a *search.* Starting with a small structure $\mathcal{F}$, we systematically explore its homomorphic images, gradually adding information. The catalog is composed of the smallest realized fragments $\mathcal{E}$ such that there is a homomorphism from $\mathcal{F}$ into $\mathcal{E}$.

Transformations and soundness. This search helps us understand 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 Π_1 to Π_2 is *sound* for a set S of goal formulas if Π_2 achieves all of the goals in S that Π_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 Π_1 to Π_2 is sound if the LTS for Π_1 simulates the LTS for Π_2, and the latter can progress whenever the former can.

To justify this as a method for showing protocol soundness, however, we must exhibit a language of security goals. We must then show that all formulas of this language, or at least all those of suitable logical form, are achieved by Π_2 if they were achieved by Π_1.

Logical form of protocol security goals. We thus 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 Φ, Ψ 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 by enrich-by-need protocol analysis, and those which—like indistinguishability results—require an essentially different method.

Protocol verification as a search. We regard the implication of Eqn. 1 as determining a search. Eqn. 1 is achieved if, starting out with fragments that satisfy Φ, and exploring their homomorphic images, we always find that Ψ must become satisfied before any realized fragment (complete execution) is encountered. Once it is satisfied, Ψ will remain satisfied in all further homomorphic images. Indeed, positive existential formulas are precisely the ones that are preserved under homomorphisms. This ensures that every complete execution that satisfies Φ will also satisfy Ψ. Thus, protocol verification may be implemented as a search through fragments, progressing via homomorphisms.

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, a 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 Π_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 [53]. 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 [62, 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 [59]; 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; this approach dates back to Casper [45]. Thus, a model-checker can examine the transition system that generates them systematically, at least for small numbers of replications of the protocol roles, seeking misbehaviors. Alternatively, for bounded numbers of sessions, one can explore the possible executions using symbolic constraints, to observe whether misbehavior is possible [2, 27, 50, 58].

Our enrich-by-need method is a variant that applies to unbounded numbers of sessions. 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 [49] and Meadows [48], 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 [60]. 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 [56], 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 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. We will formulate our contributions in more specific terms at the end of Section 1.2 (see p. 9).

1.2 Our key ideas

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

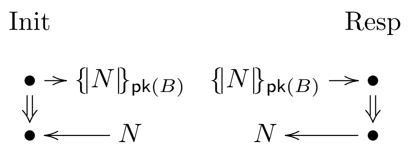

Init
Resp
$\{\!|N|\!\}_{\mathsf{pk}(B)}$
$\{\!|N|\!\}_{\mathsf{pk}(B)}$
N
N

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 [51]. It is shown with the roles written vertically 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 left half of the figure shows the initiator role, first transmitting the encrypted message for B and then receiving the freed nonce N. The right 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.

The protocol analysis process. 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, now written vertically. 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 1.3. 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 a local run of the responder. The result of adding this is shown on the right side of Fig. 2 as $\mathcal{E}$.

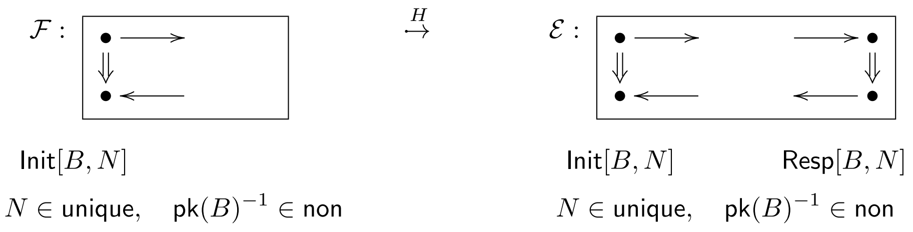

$\mathcal{F}$:
$\xrightarrow{H}$
$\mathcal{E}$:
$\mathsf{Init}[B, N]$
$N \in \mathsf{unique}$, $\mathsf{pk}(B)^{-1} \in \mathsf{non}$
$\mathsf{Init}[B, N]$
$\mathsf{Resp}[B, N]$
$N \in \mathsf{unique}$, $\mathsf{pk}(B)^{-1} \in \mathsf{non}$

Figure 2: HD Authentication Guarantee

This chain of reasoning summarizes how CPSA analyzes HD [56]. 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 $H : \mathcal{F} \mapsto \mathcal{E}$, and we will introduce a notion of homomorphism that covers it (Defn. 1.6).
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 [60, 17]), we axiomatize the relevant properties of a problem-solution LTS in Defn. 2.4.
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 $\mathsf{pk}(B)^{-1} \in \mathsf{non}$, and $N \in \mathsf{unique}$,

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

   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. 4.18 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.

$$\begin{tikzcd}
& \{\!|A\,\hat{}\,N_a|\!\}_{\mathsf{pk}(B)} & \{\!|N_a\,\hat{}\,N_b|\!\}_{\mathsf{pk}(A)} \arrow[d] & \{\!|N_b|\!\}_{\mathsf{pk}(B)} \\
\mathsf{init} & \bullet \arrow[u] \arrow[r, Rightarrow] & \bullet \arrow[r, Rightarrow] & \bullet \arrow[u] \\
\mathsf{resp} & \bullet \arrow[r, Rightarrow] & \bullet \arrow[r, Rightarrow] \arrow[d] & \bullet \\
& \{\!|A\,\hat{}\,N_a|\!\}_{\mathsf{pk}(B)} \arrow[u] & \{\!|N_a\,\hat{}\,N_b|\!\}_{\mathsf{pk}(A)} & \{\!|N_b|\!\}_{\mathsf{pk}(B)} \arrow[u]
\end{tikzcd}$$

Figure 3: Needham-Schroeder Protocol NS

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 [51] and Needham-Schroeder-Lowe [44] 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 are shown in Fig. 3, where the roles are written horizontally to save space; $a\,\hat{}\,b$ means the pair of a and b.

Lowe's corrected NSL is identical, except that the responder's name B appears in the second message, which 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. 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,

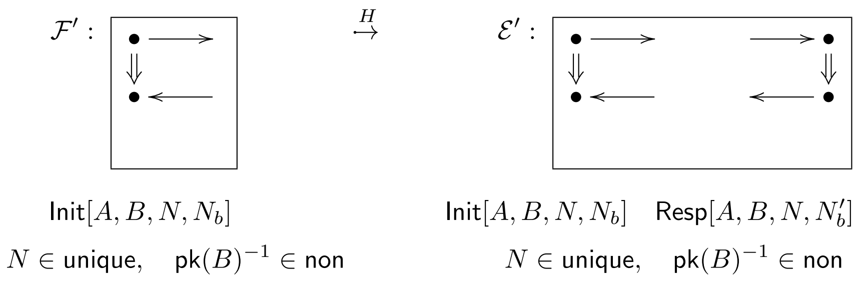

$\mathcal{F}'$:
$\overset{H}{\mapsto}$
$\mathcal{E}'$:
$\mathsf{Init}[A, B, N, N_b]$
$N \in \mathsf{unique}, \quad \mathsf{pk}(B)^{-1} \in \mathsf{non}$
$\mathsf{Init}[A, B, N, N_b]$ $\mathsf{Resp}[A, B, N, N_b']$
$N \in \mathsf{unique}, \quad \mathsf{pk}(B)^{-1} \in \mathsf{non}$

Figure 4: NS Initiator Authentication, lifting 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 Π_1 is the source protocol and Π_2 is the target protocol, we want two properties to hold:

1. When a configuration in Π_1 has a problem and therefore needs a solution, the lifted configuration in Π_2 should have a corresponding problem;

2. Given corresponding problems, each solution in Π_2 should be the lifting of some solution in Π_1.

The first says that Π_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. (See Thm. 3.9.)

Π_2 may of course pose additional problems that will also have to be solved before possible executions are found. In this case, the lower left node receives the message $\{\!|N \,\hat{}\, N_b'|\!\}_{K_A}$, while the lower right node has sent $\{\!|N \,\hat{}\, N_b|\!\}_{K_A}$, so a problem remains: Is $N_b' \neq N_b$ possible? Another problem-solution step will show that—with the assumptions shown—it is. One solution to this problem is stipulate that K_A^{-1} is compromised. The other solution is to equate $N_b' = N_b$.

The security goal language for a protocol. There is one other major layer in this paper, as motivated by characteristic 4, p. 6. 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. 4.18), and with the idea that sound protocol transformations should lift analyses (Thm. 4.23).

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.

Our contributions. We offer three contributions.

1. We consider and define enrich-by-need protocol analysis (see Def. 2.4) as a kind of labeled transition system in which the states are fragments. This LTS formalizes the activity of analysis, rather than the behavior of the protocol itself. Thm. 2.5 shows the adequacy of this view to provide a catalog of possible executions; it states that any LTS of this kind in fact covers all executions.

2. A treatment of protocol transformations (Def. 3.2) leads to Thm. 3.9. This theorem relates the LTS for analyzing the source protocol to the LTS for analyzing the target protocol. When the source LTS simulates the target LTS, with liveness for the target, then the catalog of executions in the source protocol covers the possible executions of the target (modulo the transformation).

   A strength of this treatment is that Thm. 2.5 is a special case of Thm. 3.9, namely the case in which the transformation is the identity.

3. We introduce a first order language of predicate calculus $\mathcal{GL}(\Pi)$ for each protocol Π. Authentication and confidentiality properties and many related security goals are expressed in the formulas of $\mathcal{GL}(\Pi)$ of the logical form of Eqn. 1.

   Cor. 4.18 shows that any formula of this form is achieved by a protocol Π if and only if it is satisfied in a catalog generated as in Thm. 2.5. Moreover, Thm. 3.9's condition ensures that security goal formulas are preserved by transformations (Thm. 4.23).

Thus, Defs. 2.4, 3.2 and Thms. 2.5, 3.9 form the technical core of the first half of the paper, and Thms. 4.18 and 4.23 are central to the second half.

Some related work. 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 [43, e.g.]. In the protocol composition case, roles of Π_1 also appear unchanged as roles of Π_2. Since Π_2 may also have additional roles not in the image of Π_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 [54, 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 $\Pi_1 \subseteq \Pi_2$. Their "per-goal, per-proof" criterion for protocol transformation is related to an early technique of ours [61]. 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.

Hui and Lowe [41] discuss a more restricted type of transformation, in which the structure of the roles is unchanged; a simplifying transformation simply reduces the complexity of the messages. Since their transformations are fault-preserving, their converse will be goal-preserving. However, their proof methods are very different from ours, working on traces one-by-one. Lowe and Auty [47] 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" are (generally notional) algorithms to transform their input. 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 a multi-session, active adversary [42].

Structure of this paper. After reprising the strand space terminology in Section 1.3, we devote Section 2 to clarifying the enrich-by-need approach. Protocol transformations form the focus of Section 3. Section 4 is devoted to goals and the languages we express them in; Section 4.1 gives examples, and argues that our languages are suited to expressing the bulk of properties (putting aside indistinguishability properties, which do not fit this framework). Section 4.2 introduces the first order languages of goals explicitly, and provides their semantics. Implicit in our examples are the notions of a *characteristic formula* for a fragment, and a *characteristic fragment* for a formula. We introduce these notions in Section 4.3. Finally, Section 4.4 shows that security goals are preserved under the transformations that we claim to be sound.

1.3 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 [51]. 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. We will refer to the primitive values of sort *message* as **indeterminates**; they are like algebraic indeterminates, and may be replaced by any message.

ALG populates each of these sorts with an infinite supply of primitive values.

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:

tagged pairing, the pair of t_0 and t_1 tagged with tag being written $\mathsf{tag}\ t_0\,\hat{}\,t_1$;

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 $[\![\, t_0 \,]\!]_{t_1}$.

The tags tag are chosen from some infinite set TAGS, which is disjoint from ALG. We assume that there is a distinguished tag nil, and we write $\mathsf{nil}\ t_0\,\hat{}\,t_1$ omitting the tag as $t_0\,\hat{}\,t_1$. Tags act like constants, ensuring that $\mathsf{tag1}\ t_0\,\hat{}\,t_1$ and $\mathsf{tag2}\ t_2\,\hat{}\,t_3$ never have any common instances, unless $\mathsf{tag1} = \mathsf{tag2}$. We always omit tags in examples, unless we need to prevent different syntactic units from unifying, as for instance in the first transformation in Section 3.4.

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.

For uniformity, we often regard a hash $\mathsf{hash}(t)$ as if it were the encryption 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 α are sort-respecting functions from parameters to ALG. 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 α on a message t, producing $\alpha(t)$, is defined by extending it homomorphically through the operators of t, subject to the rule that $(K^{-1})^{-1} = K$. We emphasize that

$$\alpha(\mathsf{tag1}\ t_0 \,\hat{}\, t_1) = \mathsf{tag1}\ \alpha(t_0) \,\hat{}\, \alpha(t_1),$$

meaning that substitutions always leave tags unchanged.

This algebra has the *most general unifier* (mgu) property [57]: If two messages t_0, t_1 have a common instance $\alpha(t_0) = \alpha(t_1)$, then there is a most general solution α_0. This means that $\alpha_0(t_0) = \alpha_0(t_1)$, and for every common instance $\gamma(t_0) = \gamma(t_1)$, there exists a β such that $\gamma = \beta \circ \alpha_0$.

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. $t_1 \sqsubseteq (\mathsf{tag}\ t_1 \,\hat{}\, t_2)$ and $t_2 \sqsubseteq (\mathsf{tag}\ t_1 \,\hat{}\, t_2)$;
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 $t \not\sqsubseteq \mathsf{msg}(n)$.

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 $a = b$.

Strands via Needham-Schroeder. For the sake of simplicity, we will use the familiar Needham-Schroeder [51] and Needham-Schroeder-Lowe [44] 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^{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 in Fig. 3 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. As an example, we could replace these parameters with the values C, C, N_1, and N_2; in this case, the strand would represent an execution in which a principal C intends to interact with itself as peer, although it may use different nonces in its activities as initiator and responder. 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 α that specify, for each parameter v, what value $\alpha(v)$ to replace it with.

We often label a strand with the notation $\mathsf{Init}[A, B, N_a, N_b]$, indicating what role it is an instance of, and the parameters, or what values have been substituted for the parameters. Two 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 $\bullet \overset{x}{\leftarrow}$ 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 $\bullet \overset{y}{\leftarrow}$, 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 1.1 Π *is a* protocol *iff* Π *is a finite set of strands* ρ*, 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.*

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

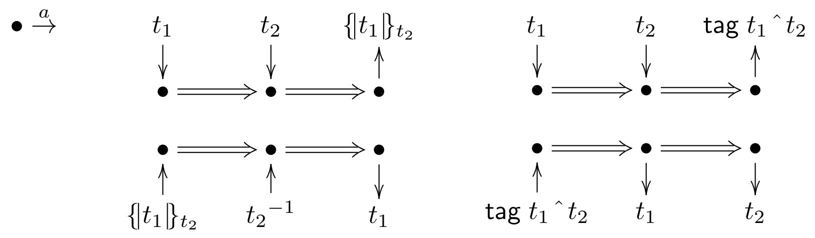


Figure 5: Adversary Roles: Generate basic value; encrypt and decrypt; concatenate and separate

We use role_unique(ρ) and role_non(ρ) in defining when a partial execution ("fragment") is permitted under the protocol Π (Defn. 1.5).

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

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. We call these *encryption* and *decryption* strands. In the middle column of Fig. 5, the upper strand is an encryption strand, and the lower strand is a decryption strand.

Adversary strands can also pair together two received messages, using any tagname tag; or, having received a tagged pair, transmit each piece separately. In the right column of Fig. 5, the upper strand is a pairing strand, and the lower strand is a separation strand.

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. We could add strands similar to those shown for these purposes. If we choose to encode those operations by encryption and pairing, then hashing and signature can be represented using the strands shown in Fig. 5.

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 $\bullet \to \bullet$ 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 1.2 *Let $\mathcal{G} = \langle \mathcal{N}, \to_G, \Rightarrow_G \rangle$ be a finite, directed acyclic graph with $E = \to_G \cup \Rightarrow_G$, where (i) $n_1 \Rightarrow_G n_2$ 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) $n_1 \to_G n_2$ 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 $n_1 \Rightarrow_G n_2$; and*
2. *If n_2 is a reception node, there exists a unique $n_1 \in \mathcal{N}$ such that $n_1 \to_G n_2$.*

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

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

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

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

Proposition 1.3 *Suppose that $\mathcal{B} = \langle \mathcal{N}, \to_G, \Rightarrow_G \rangle$ 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 $\mathcal{G} = \langle \mathcal{N}, \to_G, \Rightarrow_G \rangle$. 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.

A message t *originates at a node* n, as we mentioned before, if n is a transmission, $t \sqsubseteq \mathsf{msg}(n)$, but $t \not\sqsubseteq \mathsf{msg}(n')$ for any earlier $n' \Rightarrow^+ n$. 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.

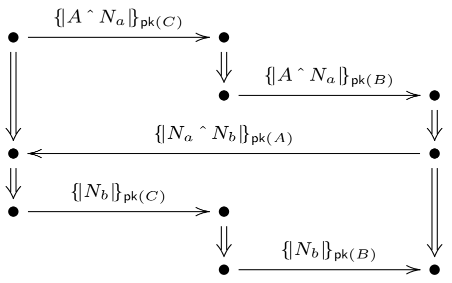

$\{\!|A \,\hat{}\, N_a|\!\}_{\mathsf{pk}(C)}$
$\{\!|A \,\hat{}\, N_a|\!\}_{\mathsf{pk}(B)}$
$\{\!|N_a \,\hat{}\, N_b|\!\}_{\mathsf{pk}(A)}$
$\{\!|N_b|\!\}_{\mathsf{pk}(C)}$
$\{\!|N_b|\!\}_{\mathsf{pk}(B)}$

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

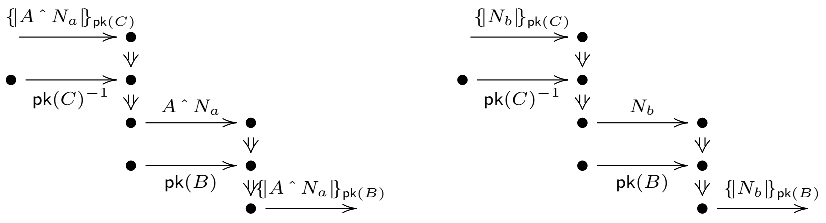

$\{\!|A \,\hat{}\, N_a|\!\}_{\mathsf{pk}(C)}$
$\mathsf{pk}(C)^{-1}$
$A \,\hat{}\, N_a$
$\mathsf{pk}(B)$
$\{\!|A \,\hat{}\, N_a|\!\}_{\mathsf{pk}(B)}$
$\{\!|N_b|\!\}_{\mathsf{pk}(C)}$
$\mathsf{pk}(C)^{-1}$
N_b
$\mathsf{pk}(B)$
$\{\!|N_b|\!\}_{\mathsf{pk}(B)}$

Figure 7: Adversary strands for Fig. 6

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.).

We use the terms *non-originating* and *uniquely originating* only in the case of basic values a, not compound values $\mathtt{tag}\ t_1 \,\hat{}\, t_2$ or $\{\!|t|\!\}_K$. Identifying non-originating keys with uncompromised ones is justified by this proposition:

Proposition 1.4 *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. [62].*

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.

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* [60], *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 1.5 *Let $\mathcal{G} = \langle \mathcal{N}, \rightarrow_G, \Rightarrow_G \rangle$ 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. $(\rightarrow_G \cup \Rightarrow_G) \subseteq \preceq$;
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.*

Let Π be a protocol. $\mathcal{F}$ is a Π-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 α and some role $\rho \in \Pi$;*
2. *if $a \in \mathsf{role\_unique}(\rho)$, and $n = \alpha(\rho) \downarrow i \in \mathsf{nodes}(\mathcal{F})$, and $\alpha(a)$ occurs in $\mathsf{msg}(n)$, then $\alpha(a) \in \mathsf{unique}(\mathcal{F})$; and*
3. *if $a \in \mathsf{role\_non}(\rho)$, and $n = \alpha(\rho) \downarrow i \in \mathsf{nodes}(\mathcal{F})$, and $\alpha(a)$ occurs in $\mathsf{msg}(n)$, then $\alpha(a) \in \mathsf{non}(\mathcal{F})$.*

Since $\Rightarrow_G$ is completely determined by $\mathcal{N}$, we will often write a fragment $\langle\langle \mathcal{N}, \rightarrow_G, \Rightarrow_G \rangle, \preceq, \mathsf{unique}, \mathsf{non} \rangle$ in the form $\langle \mathcal{N}, \rightarrow_G, \preceq, \mathsf{unique}, \mathsf{non} \rangle$. 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. Since substitutions always leave tag names tag unchanged, tag names are preserved under homomorphisms.

Definition 1.6 *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 α 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 $H\colon \mathcal{F}_1 \mapsto \mathcal{F}_2$ when H is a homomorphism from $\mathcal{F}_1$ to $\mathcal{F}_2$.

When α, α' 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, α 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 $H\colon \mathcal{F} \mapsto \mathcal{E}$.*

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

When $H = [f, \alpha]\colon \mathcal{F}_1 \mapsto \mathcal{F}_2$, 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 $H\colon \mathcal{F}_1 \mapsto_{ni} \mathcal{F}_2$*

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.

In particular, when a protocol allows *branching* behavior, it has different runs that exhibit compatible behavior up until the branch point, after which those runs may involve incompatible forms of message. In this semantics, a strand s that has not yet reached the branch point represents *both* possible kinds of extension. In particular, it may have homomorphic images in which s is mapped to instances of both of the longer roles.

1.4 Security Goals

The notion of homomorphism is central to our approach to representing security goals. A security goal such as an authentication goal may be represented by one (or a few) homomorphisms. Indeed, we can regard Fig. 2 as specifying a security goal, one which HD enforces. In particular, we can regard Fig. 2 as saying that any execution (bundle or realized fragment) $\mathcal{B}$ that contains an instance of $\mathcal{F}$ actually contains a full instance of $\mathcal{E}$.

We interpret "$\mathcal{B}$ contains an instance of $\mathcal{F}$" as meaning that there is a homomorphism $J\colon \mathcal{F} \mapsto \mathcal{B}$. The force of Fig. 2 is in fact that this homomorphism must also locate the additional structure mentioned in $\mathcal{E}$ inside $\mathcal{B}$. More precisely, the homomorphism J is compatible with H in that we can regard J as "starting off" as indicated by H, i.e. that there is a homomorphism $K\colon \mathcal{E} \mapsto \mathcal{B}$ such that J is the same as first doing H to incorporate the structure that $\mathcal{E}$ indicates must be present, and then applying K to add the rest of the information found in $\mathcal{B}$. So $J = K \circ H$.

We will say in this case that $J\colon \mathcal{F} \mapsto \mathcal{B}$ *factors through* H.

Thus, we can express a security goal in a diagram showing a homomorphism H, meaning that whenever $\mathcal{B}$ is a realized fragment and there is a homomorphism J from the source $\mathcal{F}$ of H to $\mathcal{B}$, then J must factor through H. An adversary who wants to create counterexamples to a (claimed) security goal is then trying to exhibit a realized fragment $\mathcal{B}$ and a homomorphism $J\colon \mathcal{F} \mapsto \mathcal{B}$ such that J does not factor through H. That is, it does not exhibit the structure shown in $\mathcal{E}$, embedded into $\mathcal{B}$ in any way compatible with H.

For instance, Fig. 8 expresses an authentication property that one might hope NS would enforce. 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 $\mathcal{B}$, in which there may be many runs of the protocol. Some of these may be sessions 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

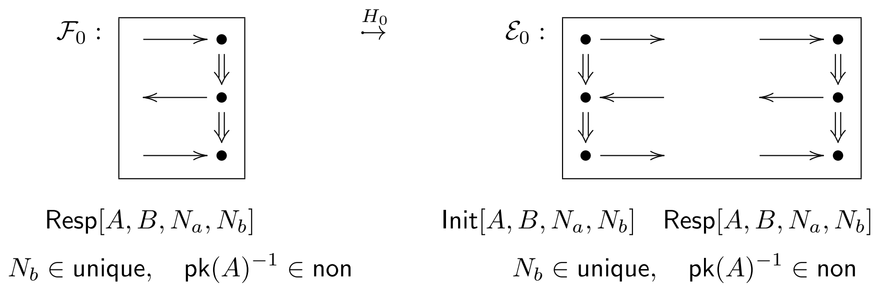

$\mathcal{F}_0$:
$\overset{H_0}{\mapsto}$
$\mathcal{E}_0$:
$\mathsf{Resp}[A, B, N_a, N_b]$
$N_b \in \mathsf{unique}, \quad \mathsf{pk}(A)^{-1} \in \mathsf{non}$
$\mathsf{Init}[A, B, N_a, N_b] \quad \mathsf{Resp}[A, B, N_a, N_b]$
$N_b \in \mathsf{unique}, \quad \mathsf{pk}(A)^{-1} \in \mathsf{non}$

Figure 8: Responder's Authentication Guarantee

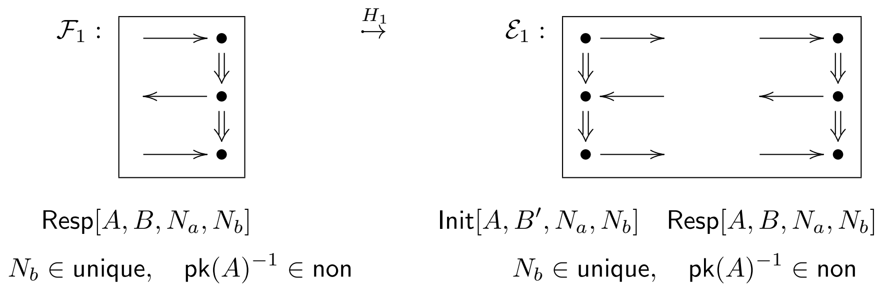

$\mathcal{F}_1$:
$\overset{H_1}{\mapsto}$
$\mathcal{E}_1$:
$\mathsf{Resp}[A, B, N_a, N_b]$
$N_b \in \mathsf{unique}, \quad \mathsf{pk}(A)^{-1} \in \mathsf{non}$
$\mathsf{Init}[A, B', N_a, N_b] \quad \mathsf{Resp}[A, B, N_a, N_b]$
$N_b \in \mathsf{unique}, \quad \mathsf{pk}(A)^{-1} \in \mathsf{non}$

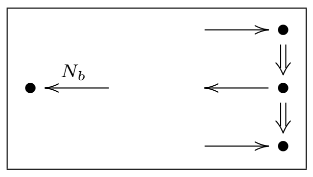
Figure 9: Responder's Weak Authentication Guarantee

sometimes freshly chosen and sometimes stale. Some sessions may be incomplete and some local runs unmatched. 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.

However, Fig. 6 shows that it does not. We may interpret Fig. 6 as a fragment $\mathcal{B}$ by specifying $\mathsf{non} = \{\mathsf{pk}(A)^{-1}\}$ and $\mathsf{unique} = \{N_b\}$. Then there is certainly a homomorphism embedding $\mathcal{F}_0$ into $\mathcal{B}$, but it does not factor through H_0. H_0 makes the two strands agree on the parameter B, whereas in $\mathcal{B}$ they do not agree on that. By contrast, $\mathcal{B}$ is not a counterexample to the weaker security goal shown in Fig. 9. Indeed, for every realized fragment in NS, any instance of the structure $\mathcal{F}_1$ is in fact part of an instance of the structure $\mathcal{E}_1$.

Many important security goals can be expressed in essentially this form. One important generalization is to specify a number of homomorphisms. Suppose that $\mathcal{F}$ is a fragment, and I is some index set, and we are given a family of homomorphisms $\{H_i\}_{i \in I}$ in which all of the homomorphisms have the same source fragment $\mathcal{F}$. Then $\{H_i\}_{i \in I}$ represents a security goal that permits a homomorphism $J \colon \mathcal{F} \mapsto \mathcal{B}$ to factor through any of the H_i. In particular, if

$\mathcal{F}$:



$\not\to$ $\cdot$

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

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

Figure 10: Secrecy for Responder's Nonce

$H_i : \mathcal{F} \to \mathcal{E}_i$, then the structure given in $\mathcal{F}$ must be present in one of the forms $\mathcal{E}_i$ in any realized fragment $\mathcal{B}$. The protocol may allow any of these situations to arise. Thus, a goal of this form is essentially disjunctive, and we will explain in Section 4 how this relates to implications with disjunctive conclusion.

In practice, the most important index set is $I = \emptyset$. In this case, the security goal says that every homomorphism $J : \mathcal{F} \to \mathcal{B}$ to a realized fragment factors through some member of the empty set, i.e. that there are no such J. That is, the behavior shown in $\mathcal{F}$ can never occur in any execution. This corresponds to the fact that the disjunction with zero disjuncts is the constant, *falsehood*.

We often express secrecy goals this way, but showing some behavior together with a listener strand $\bullet \overset{N}{\leftarrow}$. If this cannot be enriched to form a complete execution, that means that the behavior in question is incompatible with N being disclosed on the network. Thus, N will remain secret. The NSL secrecy goal for the responder's nonce N_b is shown in Fig. 10. By the symbol $\not\to \cdot$, we mean to convey the empty family, i.e. that there is no realized fragment $\mathcal{D}$ such that $\mathcal{F} \to \mathcal{D}$.

Many security goals may be expressed in this form. They include such properties as recency, in which the target fragment contains ordering information; implicit authentication [8, 24], in which the source fragment contains multiple strands, and the target equates some of their parameters; forward secrecy in two flavors [24]; and a variety of injective agreement [46], in which multiple sessions of the same role are equated.

We summarize this treatment of security goals in a definition:

Definition 1.7 *1.* $\{H_i\}_{i \in I}$ *is a* family of homomorphisms based in $\mathcal{F}$ *iff, for each $i \in I$, $H_i : \mathcal{F} \to \mathcal{E}_i$ is a homomorphism from the same source fragment $\mathcal{F}$.*

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

Π enforces $\{H_i\}_{i \in I}$ *iff every homomorphism from $\mathcal{F}$ to a realized* Π-*fragment $\mathcal{D}$ factors through some H_i.*

That is Π enforces $\{H_i\}_{i\in I}$ iff, for every realized Π-fragment $\mathcal{B}$ and homomorphism $J\colon \mathcal{F} \mapsto \mathcal{B}$, there exists an $i \in I$ and a homomorphism $K\colon \mathcal{E}_i \mapsto \mathcal{B}$ such that $J = K \circ H_i$.

In Section 4.2, we will give an alternative but compatible account of security goals as first order logical formulas, specifically as universally quantified implications of the form of Eqn. 1 with positive existential antecedent and conclusion.

2 Enrich-by-Need Protocol Analysis

The *enrich-by-need* idea in protocol analysis is a method for exploring the possible executions of a protocol. One starts with some fragment of interest, $\mathcal{F}_0$. One then explores its homomorphic images by taking "small steps," i.e. homomorphisms that add a small amount of information at a time. The purpose of the exploration is to find realized fragments; these realized fragments show "what can happen" in executions that have a portion of the form given in $\mathcal{F}_0$.

2.1 Forms of Search

There are two main approaches to generating cohorts. The first introduces adversary strands. The second avoids them entirely, and works only with skeletons, i.e. fragments containing no adversary strands. To formulate the two approaches, we will use the notion of a *component*:

Definition 2.1 *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 and* tag *such that $t_1 = \mathsf{tag}\ t_2 \,\hat{}\, t_3$ 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 [60, 17], and related to NPA [48], 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 α 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 some larger encrypted unit e. This is relevant only when e is an ingredient in some message transmitted on a regular node of some Π-strand [52, 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]. This approach uses only *skeletons*, meaning fragments that are free of adversary actions. A skeleton also eschews communication edges $\rightarrow$, using only the precedence order and the strand relation $\Rightarrow$ to constrain the relations among nodes:

Definition 2.2 *A fragment* $\mathcal{F} = \langle \mathcal{G}, \preceq, \mathsf{unique}, \mathsf{non} \rangle$ *is a* skeleton *iff* $\mathsf{nodes}(\mathcal{G}) = \mathsf{regnodes}(\mathcal{G})$ *and* $\rightarrow_{\mathcal{G}} = \emptyset$.

$\mathcal{F} = \langle \mathcal{G}, \preceq, \mathsf{unique}, \mathsf{non} \rangle$ *is a* realized skeleton *iff there exists a* $\mathcal{G}'$ *such that* $\mathsf{regnodes}(\mathcal{G}) = \mathsf{regnodes}(\mathcal{G}')$ *and* $\mathcal{F}' = \langle \mathcal{G}', \preceq, \mathsf{unique}, \mathsf{non} \rangle$ *is a realized fragment. That is,* $\mathcal{F}'$ *differs from* $\mathcal{F}$ *only in adversary nodes and* $\rightarrow$.

We generally write $\mathbb{A}, \mathbb{B}$, etc. for skeletons.

Here we consider a basic value $c \in \mathsf{unique}(\mathbb{A})$ or else an encryption c. We choose a reception node $n_1 \in \mathbb{A}$ such that $c \sqsubseteq \mathsf{msg}(n_1)$. We also choose a set of encryptions S. Suppose:

- If $\{|t|\}_K \in S$, then K^{-1} is not a component of any node $m \preceq_{\mathbb{A}} n_1$.

- If $n_0 \prec_{\mathbb{A}} n_1$, 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 never enters the key of any encryption. (We say that c is found outside S in n_1.)

Under these conditions, n_1, c, S is an *unsolved authentication test* in $\mathbb{A}$. The "test" is to explain how c got outside the encryptions S, as it 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 skeleton $\mathbb{A}$ in the following ways:

1. If there is a substitution α such that $\alpha(c)$ is found only within $\alpha(S)$ in $\alpha(\mathsf{msg}(n_1))$, then applying α makes the test disappear.

2. Adding a listener node for a key K^{-1} such that $\{|t|\}_K \in S$ to $\mathbb{A}$ 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 [56] to see how this idea is implemented in the protocol tool CPSA. In this method, the problems are the unsolved authentication tests.

2.2 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 ℓ in $\mathcal{F}$, i.e. either ℓ is a pair n_1, c, where c is a component received on n_1 but not transmitted earlier, or ℓ is an unsolved test n_1, c, S, a *cohort* for ℓ is a family of homomorphisms $\{H_i\}_{i \in I}$ based in $\mathcal{F}$ such that:

- If $H_i : \mathcal{F} \mapsto \mathcal{E}_i$, then the image of ℓ is solved in $\mathcal{E}_i$.

- If $K : \mathcal{F} \mapsto \mathcal{D}$, where $\mathcal{D}$ is realized, then $K = J \circ H_i$ for some J and $i \in I$.

A cohort for ℓ is thus a set of maximally general ways of solving ℓ.

As a special case, a cohort for ℓ could be the empty set of homomorphisms. 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 fragment.

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 ℓ in it. If there is none, we can immediately construct a realized fragment from it. Otherwise, we choose a problem ℓ 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. 4.17 to this family to decide whether a goal G is achieved. We may also apply Cor. 4.19 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 ϕ is recursively enumerable. Thus, the relation Π *achieves* G is co-r.e.

2.3 Axiomatizing Enrich-by-need

Homomorphisms determine a preorder, not a partial order, since $J \circ H$ is not always an isomorphism when $\mathcal{F} \overset{H}{\mapsto} \mathcal{E} \overset{J}{\mapsto} \mathcal{F}$. 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 $H \colon \mathcal{F} \mapsto_{ni} \mathcal{E}$ determine a partial order $\leq_{ni}$ on fragments to within isomorphism.

Lemma 2.3 ([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 $\mapsto$. Skeletons $\mathbb{A}_0$ determine starting points for the search; protocol analysis then seeks realized fragments $\mathcal{D}$ such that $\mathbb{A} \mapsto \mathcal{D}$. 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* [56]. Within the set of all realized $\mathbb{B}$ such that $\mathbb{A} \mapsto \mathbb{B}$, 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 ℓ, 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 $\leq_{ni}$. CPSA's search separates into two phases. After an initial non-node-injective step, all of its test-solving takes place in the node-injective ordering (see [37, Thm. 6.5]). Scyther's steps are also node-injective.[1]

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, as we will illustrate in an example below (Section 2.4).

We let S be a set of fragments and Λ 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 ℓ 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 ℓ of these forms,

[1]Personal communication with C. Cremers.

$\{\mathcal{E}_i \colon \mathcal{F} \overset{\ell}{\rightsquigarrow} \mathcal{E}_i\}$ defines the cohort solving the problem ℓ. 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 2.4 *Suppose given S, a set of fragments, $\mathsf{dead} \in \Lambda$, and a ternary relation $\cdot \overset{\cdot}{\rightsquigarrow} \cdot \subseteq S \times \Lambda \times S$ 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 $J = \mathcal{F} \overset{H}{\to} \mathcal{E}' \overset{K}{\to} \mathcal{D}$;*
   - *(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 σ, then there is a definite $H \colon \mathcal{F} \mapsto \mathcal{E}$ corresponding to σ.

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_σ 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 [56, 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 hundreds.

We often use CPSA on a number of starting points; it is effectively constructing a PSLTS containing that set of skeletons and those generated from them by a sequence of tests.

Theorem 2.5 *Suppose that $\cdot \overset{\cdot}{\rightsquigarrow} \cdot$ 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 Ξ be the set of all sequences

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

starting at $\mathcal{F}_0$ such that for each $i \le k$, $\mathcal{F}_i \mapsto \mathcal{D}$. $\Xi \neq \emptyset$ because it contains the trivial sequence $\mathcal{F}_0$. By Def. 2.4, Clause 3b, there are maximal sequences in Ξ. So let $p = \mathcal{F}_0 \overset{\ell_1}{\rightsquigarrow} \mathcal{F}_1 \rightsquigarrow \cdots \rightsquigarrow \mathcal{F}_{k-1} \overset{\ell_k}{\rightsquigarrow} \mathcal{F}_k$ be maximal.

We claim that $\mathcal{F}_k$ is realized. Otherwise, there must be some transition $\mathcal{F}_k \overset{\ell}{\rightsquigarrow} \mathcal{C}$. 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-dead label ℓ such that $\mathcal{F}_k \overset{\ell}{\rightsquigarrow} \cdot$. Hence, by clause 3a, there exists $\mathcal{E}'$ s.t. $\mathcal{F}_k \overset{\ell}{\rightsquigarrow} \mathcal{E}'$, and $\mathcal{F} \mapsto \mathcal{E}' \mapsto \mathcal{D}$. Hence, $p \overset{\ell}{\rightsquigarrow} \mathcal{E}' \in \Xi$, contradicting the maximality of p. $\square$

Thm. 2.5 is related to the view of security goals that we have described in Section 1.4, summarized in Defn. 1.7. It suggests a semi-algorithm for checking whether a protocol enforces a goal.

In particular, suppose that we are interested in a security goal $\mathcal{G} = \{H_i\}_{i \in I}$, which is a family of homomorphisms based in $\mathcal{F}$. We can check whether $\mathcal{G}$ holds by exploring a PSLTS that includes $\mathcal{F}$. As the search reaches any realized fragment $\mathcal{D}$, we have to check whether the map $\mathcal{F} \rightsquigarrow^* \mathcal{D}$ factors through at least one of the H_i. If not, we have obtained a counterexample to the goal, namely the realized skeleton $\mathcal{D}$. Otherwise, we continue. If we complete the exploration of the fragments accessible from $\mathcal{F}$ within the PSLTS without finding a counterexample, then the protocol enforces the goal.

2.4 Some example transition systems

As an example, consider the protocol HD. We may take the set of fragments S to contain the two skeletons $\mathcal{F}, \mathcal{E}$ shown in Fig. 2, p. 5. The relation $\cdot \overset{\cdot}{\rightsquigarrow} \cdot$ contains the a triple $\mathcal{F} \overset{\ell}{\rightsquigarrow} \mathcal{E}$. CPSA represents this ℓ with the unexplained, lower node of $\mathcal{F}$, the fresh nonce N, and the encryption $\{\!|N|\!\}_{\mathsf{pk}(B)}$ that N was transmitted within. This tiny, two-fragment PSLTS satisfies the axioms.

We consider in more detail a PSLTS which includes the starting points for authentication for the NSL initiator and responder (Figs. 11–12). Each starting point consists of a skeleton ($\mathbb{A}$, $\mathbb{D}$) which contains a strand of that role, together with the assumption that the nonce originating on that role is uniquely originating. In the case of the responder, we assume that the *initiator's* private key is non-originating; in the case of the initiator, the natural assumption is that *both* private keys are non-originating [62]. This set S of skeletons contains the six members $\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{C}', \mathbb{D}, \mathbb{E}$; its transition relation $\rightsquigarrow$ contains only four transitions. Indeed, in this analysis, there are no dead transitions. The cohort $\rightsquigarrow \mathbb{B}\ell_2\mathbb{C}, \mathbb{C}'$ contains two transitions, while the cohorts for ℓ_1 and ℓ_3 are each singletons. This PSLTS represents the NSL authentication analyses, as analysis with CPSA confirms.[2] One can regard CPSA as computing this finite PSLTS as a

[2] CPSA in fact eliminates $\mathbb{C}'$. It adds nothing, because the homomorphism $\mathbb{A} \mapsto \mathbb{C}'$ actually factors through $\mathbb{A} \mapsto \mathbb{C}$. See URL http://web.cs.wpi.edu/~guttman/transformations/nsl.xhtml for the analysis.

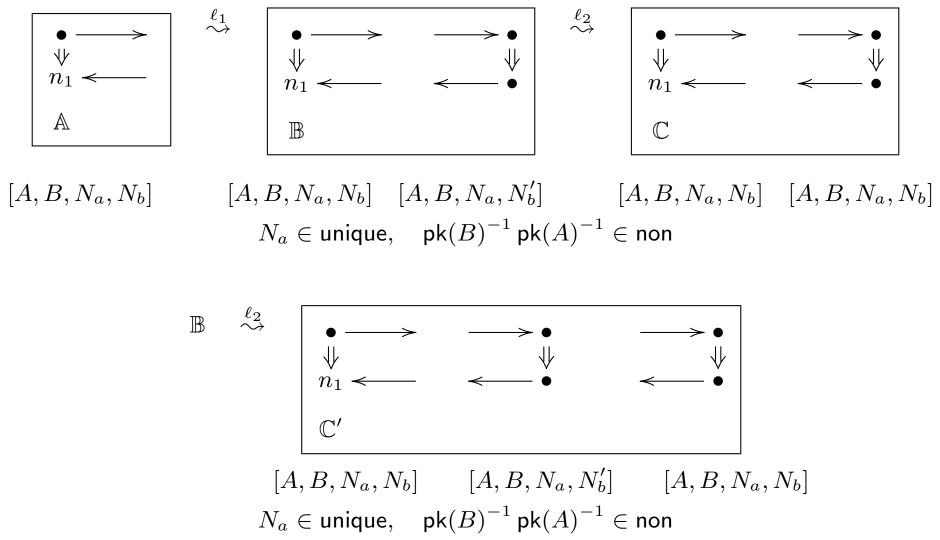

n_1
$\mathbb{A}$
ℓ_1
$\mathbb{B}$
ℓ_2
$\mathbb{C}$
$[A, B, N_a, N_b]$
$[A, B, N_a, N_b]$ $[A, B, N_a, N_b']$
$[A, B, N_a, N_b]$ $[A, B, N_a, N_b]$
$N_a \in$ unique, $\mathsf{pk}(B)^{-1}\,\mathsf{pk}(A)^{-1} \in$ non
$\mathbb{B}$
ℓ_2
$\mathbb{C}'$
$[A, B, N_a, N_b]$ $[A, B, N_a, N_b']$ $[A, B, N_a, N_b]$
$N_a \in$ unique, $\mathsf{pk}(B)^{-1}\,\mathsf{pk}(A)^{-1} \in$ non

Figure 11: NSL Authentication for the initiator

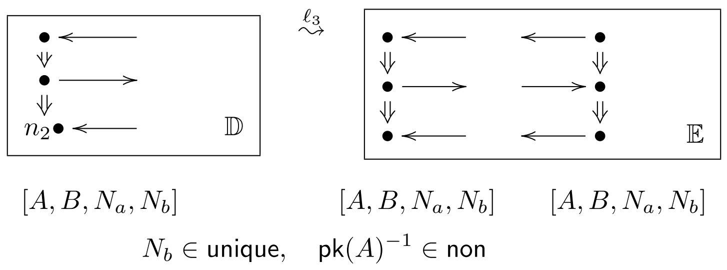

n_2
$\mathbb{D}$
ℓ_3
$\mathbb{E}$
$[A, B, N_a, N_b]$
$[A, B, N_a, N_b]$ $[A, B, N_a, N_b]$
$N_b \in$ unique, $\mathsf{pk}(A)^{-1} \in$ non

Figure 12: NSL Authentication for the responder

Label	Critical value	Node	Escape set members
ℓ_1	N_a	n_1	$\{\!\|A \mathbin{\hat{}} N_a\|\!\}_{\mathsf{pk}(B)}$
ℓ_2	N_a	n_1	$\{\!\|A \mathbin{\hat{}} N_a\|\!\}_{\mathsf{pk}(B)}$, and $\{\!\|N_a \mathbin{\hat{}} N \mathbin{\hat{}} B\|\!\}_{\mathsf{pk}(A)}$, for all $N \neq N_b$
ℓ_3	N_b	n_2	$\{\!\|N_a \mathbin{\hat{}} N_b \mathbin{\hat{}} B\|\!\}_{\mathsf{pk}(A)}$

Table 1: Labels for the NSL authentication PSLTS

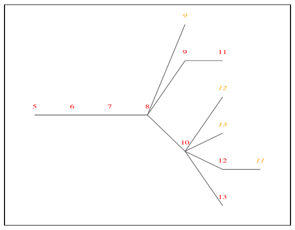


Figure 14: LTS CPSA generates analyzing Fig. 13

substructure of the infinite PSLTS of all NSL skeletons, tests, and their solution cohorts.

The labels ℓ_i are notationally a bit complex. We present them in Table 1. Labels ℓ_1 and ℓ_2 are both explaining how N_a can be received in node n_1. In ℓ_1, which concerns $\mathbb{A}$, N_a has been sent only in the single form $\{\!|A \mathbin{\hat{}} N_a|\!\}_{\mathsf{pk}(B)}$, whereas in $\mathbb{B}$ it has also been transmitted inside the encryption $\{\!|N_a \mathbin{\hat{}} N_b' \mathbin{\hat{}} B|\!\}_{\mathsf{pk}(A)}$. Label ℓ_2 uses as its escape set $\{\!|A \mathbin{\hat{}} N_a|\!\}_{\mathsf{pk}(B)}$ together with *all* messages of the form $\{\!|N_a \mathbin{\hat{}} N \mathbin{\hat{}} B|\!\}_{\mathsf{pk}(A)}$, for any choice $N \neq N_b$ for the nonce N. This escape set ensures that the protocol actions that transport N_a into useful new messages in fact associate it with the desired nonce N_b in the message $\{\!|N_a \mathbin{\hat{}} N_b \mathbin{\hat{}} B|\!\}_{\mathsf{pk}(A)}$. CPSA does not work with infinite escape sets such as this one, but retains information about the original starting point instead. This causes it to provide essentially the same results.

Label ℓ_3 concerns N_b, and explaining how it can be received in n_2 of $\mathbb{D}$, when it has only been sent in the form $\{\!|N_a \mathbin{\hat{}} N_b \mathbin{\hat{}} B|\!\}_{\mathsf{pk}(A)}$.

Naturally, this PSLTS does not tell us everything that we might like to know about NSL; for instance, it says nothing about secrecy. Other questions may be resolved by using CPSA to construct other PSLTSs. They may be larger; for instance, the PSLTS CPSA constructs for the initiator's secrecy goal contains nine distinct skeletons, and the one for the responder's secrecy goal contains five. Fig. 13 shows the starting point for the search for the initiator's secrecy goal. The first node of an initiator strand uses the nonce N_a, subject to the assumptions that $N_a \in \mathsf{unique}$ and $K_A^{-1}, K_B^{-1} \in \mathsf{non}$. It also

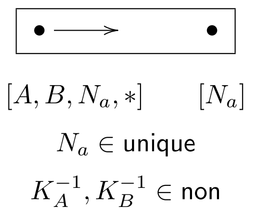

$[A, B, N_a, *]$
$[N_a]$
$N_a \in \mathsf{unique}$
$K_A^{-1}, K_B^{-1} \in \mathsf{non}$

Figure 13: Initiator's secrecy query

contains a listener node that purportedly receives N_a without cryptographic protection. CPSA, in searching for all ways to realize this starting point, encounters empty cohorts which entail that that the starting point is unrealizable. The graph of CPSA's search steps appears in Fig. 14. Italicized terminal nodes are duplicates of nodes explored elsewhere. The terminal nodes shown in red (when printed in color) are dead skeletons $\mathbb{A} \overset{\mathsf{dead}}{\rightsquigarrow} \mathbb{A}$.

CPSA can perform successful analyses exploring hundreds or sometimes thousands of skeletons.

3 Protocol Transformations

Protocol design is an art of reuse. A few basic patterns for achieving authentication and confidentiality—despite actively malicious parties—are adapted to many 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. This explanation is contained in Thm. 3.9. It concerns 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. It in turn will lead in Section 4.4 to Thm. 4.23; that results justifies inferring that a transformed protocol satisfies some security goals, in the sense of logical formulas, when the source protocol did.

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 ρ that it instantiates, and the substitution α such that $s = \alpha(\rho)$, as in Defn. 3.4. 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 3.1). We then prove that transformations yield full and faithful functors on skeletons (Section 3.2). In Section 3.3, we give Thm. 3.9, generalizing Thm. 2.5. Some examples of sound transformations that piggyback messages follow in 3.4.

3.1 Some Protocol Transformations

We have already described the protocol HD, defined in Fig. 1. 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 Π_1 and target protocol Π_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 ρ_1 with a node along ρ_2; and
- sends each parameter a of role ρ_1 to a parameter b of ρ_2.

Evidently the nodes must be associated in a way that preserves their order, since otherwise an execution of the source protocol might not correspond to an execution of the target. Also, a transmission node in the source protocol must correspond to a transmission node in the target protocol, and likewise for receptions. Otherwise, again, executions will remain meaningful.

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.

$$\begin{tikzcd}
\text{QAf} & \bullet \arrow[r, Rightarrow] \arrow[d] & \bullet & \bullet \arrow[r, Rightarrow] \arrow[d] & \bullet & \text{QNg} \\
& \{\!|Q\,\hat{}\,Y\,\hat{}\,N|\!\}_{\mathsf{pk}(A)} & Y \arrow[u] & \{\!|Q\,\hat{}\,Y\,\hat{}\,N|\!\}_{\mathsf{pk}(A)} & N \arrow[u] & \\
& \{\!|Q\,\hat{}\,Y\,\hat{}\,N|\!\}_{\mathsf{pk}(A)} \arrow[d] & Y & \{\!|Q\,\hat{}\,Y\,\hat{}\,N|\!\}_{\mathsf{pk}(A)} \arrow[d] & N & \\
\text{AnAf} & \bullet \arrow[r, Rightarrow] & \bullet \arrow[u] & \bullet \arrow[r, Rightarrow] & \bullet \arrow[u] & \text{AnNg}
\end{tikzcd}$$

Figure 15: The Yes-or-No Protocol YN

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.

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 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.

This example illustrates the use of a substitution γ to correlate the right parameters of the two protocols. Different substitutions are needed in different transformation, even if they have the same source and target protocols.

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.

There is a transformation from YN to YN^+ in which we map AnAf and AnNg to the two roles shown on the left side and QAf and QNg to the

$$\begin{tikzcd}
& A \arrow[rrr, "\{\!|R\,\hat{}\,A|\!\}_B"] \arrow[d, Rightarrow] & & & B \arrow[d, Rightarrow] & \\
& \bullet \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & & & \bullet \arrow[lll, "\{\!|R\,\hat{}\,Q\,\hat{}\,Y\,\hat{}\,N\,\hat{}\,B|\!\}_A"'] \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\
\bullet \arrow[rrr, bend right, "Y"'] & & \bullet \arrow[rrr, bend right, "N"'] & \bullet & & \bullet
\end{tikzcd}$$

Figure 16: Answering Questions with YN^+

two on the right side. The two nodes of each source protocol role are mapped to the second and third node of the YN^+ roles. There is a transformation from HD directly to 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 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 for the NSL secrecy goals, because YN^+ discloses B's nonces, which NSL does not.

3.2 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 3.1 *A substitution γ is* suitable *for ρ_1, ρ_2 iff for some set X such that $\mathsf{Params}(\rho_1) \subseteq X$, γ is a bijection between X and $\mathsf{Params}(\rho_2)$.*

So, γ is injective going forward from parameters of ρ_1, and on a set of other values that it maps to the remaining parameters of ρ_2. When γ is suitable, we can apply its inverse to ρ_2; if e.g. $\alpha(\rho_1)$ is an instance of ρ_1, then $\alpha(\gamma^{-1}(\rho_2))$ is a corresponding instance of ρ_2.

Definition 3.2 (Transformation) *Suppose T maps each role $\rho_1 \in \Pi_1$ to a triple ρ_2, g, γ, where $\rho_2 \in \Pi_2$, $g \colon \mathbb{N}^+ \to \mathbb{N}^+$, and γ is a substitution suitable for ρ_1, ρ_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 to role ρ_1 and $j \leq g(i)$:*
   (a) if $\gamma(x)$ originates on $\rho_2 \downarrow j$, then there exists a $k \leq i$ such that x originates on $\rho_1 \downarrow k$; and
   (b) if $\gamma(x) \sqsubseteq \mathsf{msg}(\rho_2 \downarrow j)$, then there exists a $k \leq i$ such that $x \sqsubseteq \mathsf{msg}(\rho_1 \downarrow k)$.
4. *Let $\rho_1, \sigma_1 \in \Pi_1$, let $T(\sigma_1) = \sigma_2, h, \delta$, and let α, β 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 α', β' that agree with α, β on the parameters of ρ_1, σ_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 Π_1 to Π_2, we write $T \colon \Pi_1 \to \Pi_2$.

The first two clauses say that T should *preserve* order and direction (send vs. receive). The third clause says that T should *reflect* where parameters originate and where they are ingredients.

The last clause says that when the same strand can be viewed (up to height i) as an instance of either ρ_1 or σ_1, then the transformation handles it the same no matter which way we view it. In strand spaces, branching behaviors are represented in this way, namely as roles that agree up to the branch point, but disagree thereafter. This clause ensures that commitment to one branch cannot occur earlier in the target of a transformation that it does in the source.

These conditions are preserved under composition:

Lemma 3.3 *Suppose that $T_1 \colon \Pi_1 \to \Pi_2$ and $T_2 \colon \Pi_2 \to \Pi_3$. Let us write $T_2 \circ T_1$ for the function that maps $\rho_1 \in \Pi_1$ to*

$$\rho_3,\ (g_3 \circ g_2),\ (\gamma_3 \circ \gamma_2),$$

where $T_1(\rho_1) = \rho_2, g_2, \gamma_2$ and $T_2(\rho_2) = \rho_3, g_3, \gamma_3$. Then $T_2 \circ T_1 \colon \Pi_1 \to \Pi_3$.

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 ρ and substitution α. Thus, we can always represent the nodes of a skeleton as a set of triples ρ, α, 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 α, α' differ only on parameters that do not appear in $\mathbb{A}$, then replacing ρ, α, i by ρ, α', i in a role-substitution representation of $\mathsf{nodes}(\mathbb{A})$ yields another role-substitution representation of $\mathsf{nodes}(\mathbb{A})$.

Definition 3.4 *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 γ to find how to apply α to ρ_2; we then take the $g(j)^{\text{th}}$ node of the resulting strand.

Hence, if S is a set of triples ρ, α, 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 Π_1-skeleton, and let $\mathbb{C}$ be a Π_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 3.5 *Let* $\mathbb{A}$ *be a* Π_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*

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

Let a be a parameter of ρ_2, and b be a parameter of σ_2 that are not in the images of the parameters of ρ_1 under γ and σ_1 under δ. Then:

   (a) *$\alpha(a)$ and $\beta(b)$ are both parameters;*
   (b) *$\alpha(a) = \beta(b)$ implies ρ_2 and σ_2 are the same, and a and b are also same.*

Proof. **1.** For each strand $s \in \mathsf{strands}(\mathbb{A})$, letting $s = \alpha(\rho_1)$, construct $\mathbb{C}_0$ by selecting an instance $\lambda(s) = \beta_s(\rho_2)$, of height $g(i)$, where the instantiation β_s associates not-yet-used parameters with each parameter of ρ_2. Let $\mathsf{unique}(\lambda(s)) = \beta_s(\mathsf{role\_unique}(\rho_2))$ be the role-unique of ρ_2 under β_s, and let $\mathsf{non}(\lambda(s)) = \beta_s(\mathsf{role\_non}(\rho_2))$ be the role-non of ρ_2 under β_s. Define $\mathsf{unique}_{\mathbb{C}_0}, \mathsf{non}_{\mathbb{C}_0}$ to be the minimal possible sets, i.e.

$$\begin{aligned} \mathsf{unique}_{\mathbb{C}_0} &= \bigcup_{s \in \mathbb{A}} \mathsf{unique}(\lambda(s)) \\ \mathsf{non}_{\mathbb{C}_0} &= \bigcup_{s \in \mathbb{A}} \mathsf{non}(\lambda(s)). \end{aligned}$$

Define $n_0 \preceq_{\mathbb{C}_0} n_1$ iff $n_0 \Rightarrow^+ n_1$.

The resulting object $\mathbb{C}_0$ is a Π_2-skeleton: The conditions on the nodes and ordering are immediate from the definitions. Moreover, Definition 3.2, Clause 3b ensures that any node in $\mathbb{C}_0$ in which $a \in \mathsf{non}_{\mathbb{C}_0}$ appears as an ingredient is an image of a node in $\mathbb{A}$ in which it already appeared as an ingredient. Thus, the condition on non-origination holds is satisfied in $\mathbb{C}_0$ because it was satisfied in $\mathbb{A}$. Similarly, Definition 3.2, Clause 3a ensures that any node in $\mathbb{C}_0$ in which $a \in \mathsf{unique}_{\mathbb{C}_0}$ originates is an image of a node in $\mathbb{A}$ in which it already originated. Thus, the unique origination condition holds in $\mathbb{C}_0$ because it held in $\mathbb{A}$.

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

Using these homomorphisms, apply Lemma 4.15, clause 3 repeatedly, once for each pair of nodes $\lambda(n), \lambda(m)$ such that $n \preceq_{\mathbb{A}} m$. Use Lemma 4.15, 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 ρ_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 4.15, clause 1, which affect only the parameters $\gamma(c)$ where c is a parameter of ρ_1. $\square$

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 3.6 *$T(\mathbb{A})$ 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 3.7 *Let $T \colon \Pi_1 \to \Pi_2$.*

1. *If $H \colon \mathbb{A} \mapsto \mathbb{B}$ is a Π_1-homomorphism, there is a Π_2-homomorphism $G \colon T(\mathbb{A}) \mapsto T(\mathbb{B})$ 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 $G \colon T(\mathbb{A}) \mapsto T(\mathbb{B})$, then there is an $H \colon \mathbb{A} \mapsto \mathbb{B}$ 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 ρ_1 and α, 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

$$\begin{tikzcd} T(\mathbb{A}) \arrow[r, "G"] & T(\mathbb{B}) \\ \mathbb{A} \arrow[u, "\mathsf{lift}_T"] \arrow[r, "H"] & \mathbb{B} \arrow[u, "\mathsf{lift}_T"'] \end{tikzcd}$$

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

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. $\square$

3.3 Preservation via PSLTSs

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

To show that the goals G_Φ are preserved under a transformation $T \colon \Pi_1 \to \Pi_2$, we would like to exhibit a PSLTS for Π_1 that contains $\mathsf{cfrag}(\Phi)$, and a PSLTS for Π_2 that contains $T(\mathsf{cfrag}(\Phi))$. If these PSLTSs match up, then all the goals G_Φ 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. 2.5, at least for skeletons; Thm. 2.5 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 Λ_1, Λ_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 3.8 *Let $\Delta \colon \Lambda_1 \to \Lambda_2$ be a function between the Λ_i, and let $T \colon \Pi_1 \to \Pi_2$ be a protocol transformation. Let $\rightsquigarrow_1$ be a* PSLTS *on Π_1 using labels Λ_1; let $\rightsquigarrow_2$ be a* PSLTS *on Π_2 using labels Λ_2.*

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

Let Δ be a relabeling function.

2. *T, Δ* 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, Δ 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 3.9 (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 Π_1 and Π_2 resp., with $\mathbb{A}_0 \in S(\rightsquigarrow_1)$ and $T(\mathbb{A}_0) \in S(\rightsquigarrow_2)$. Suppose that:*

1. *T, Δ preserve progress for $\rightsquigarrow_1, \rightsquigarrow_2$;*

2. *$\rightsquigarrow_1$ simulates $\rightsquigarrow_2$ under T, Δ.*

$$\begin{tikzcd}
T(\mathbb{A}_0) \arrow[r, "K"'] \arrow[rr, bend left, "H"] \arrow[d, dash, dotted] & T(\mathbb{A}_k) \arrow[r, "J"'] \arrow[d, dash, dotted] & \mathbb{C} \\
\mathbb{A}_0 \arrow[r, squiggly] & \mathbb{A}_k &
\end{tikzcd}$$

For every Π_2-realized $\mathbb{C}$, if $H \colon T(\mathbb{A}_0) \to \mathbb{C}$, there is a Π_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 Σ be the set of $\rightsquigarrow_1$ paths σ 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 Σ_H be the set of $\sigma \in \Sigma$ such that, for some $J\colon T(\mathbb{A}_j) \mapsto \mathbb{C}$, we have $H = J \circ H_\sigma$. These σ are the ones that never diverge from H. Σ_H is non-empty since the empty path is in it.

By Defn. 2.4, Clause 3b, there are maximal members in Σ_H, i.e. σ_1 such that no extension $\sigma_1^\frown\ell$ is in Σ_H.

So let σ_1 be maximal in Σ_H, where $\mathbb{A}_0 \overset{\sigma_1}{\rightsquigarrow}_1^* \mathbb{A}_j$. By construction, $T(H_{\sigma_1})\colon T(\mathbb{A}_0) \mapsto T(\mathbb{A}_j)$, 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. 2.4, Clause 1, there is a label ℓ 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. 2.4, Clause 2 contradicts the assumption that $J\colon T(\mathbb{A}_j) \mapsto \mathbb{C}$.

Using Clause 3a, there is in fact a $\mathcal{E}'$ such that $T(\mathbb{A}_j) \overset{\Delta(\ell)}{\rightsquigarrow}_2 \mathcal{E}'$ and $J = T(\mathbb{A}_j) \overset{H_\ell}{\mapsto} \mathcal{E}' \overset{K}{\mapsto} \mathcal{C}$. That is, $H_{\Delta(\ell)}\colon T(\mathbb{A}_j) \mapsto \mathcal{E}'$ 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 Σ_H, contradicting the maximality of σ. Hence, $\mathbb{A}_j$ is realized. $\square$

The proof of Thm. 2.5 is essentially this proof, letting T, Δ be the identity.

3.4 Example: Piggybacking Secrets

We now illustrate the power of this result by providing a generic transformation, i.e. an infinite family of transformations. It uses NSL as a key agreement method to offer authentication and confidentiality services to transport a payload message v_0 between the participants. The payload here is a simple, basic data value. We offer two versions of this protocol, one (which we will call Π^i) in which the initiator sends v_0 to the responder, and another (Π^r) in which the responder sends it to the initiator. These protocols inherit the authentication properties of NSL, and secrecy for the nonces N_a, N_b. They provide secrecy for v_0 by sending it encrypted, using $\mathsf{hash}(N_a \,\hat{}\, N_b)$ as the key. The association of v_0 with N_a, N_b ensures its authenticity also.

We then consider the protocols Π^i_t, Π^r_t that result when we insert a message t in place of v_0. The message t may contain many simple data values instead of just one. It may also package them in any message format, e.g. with nested encryptions. We limit our claims here to the case where t does not contain any nonces (which might unify with the nonces N_a, N_b of the key agreement phase).

$$\begin{tikzcd}
\{\!|A \,\hat{}\, N_a|\!\}_{\mathsf{pk}(B)} & \{\!|N_a \,\hat{}\, N_b \,\hat{}\, B|\!\}_{\mathsf{pk}(A)} \arrow[d] & \{\!|N_b|\!\}_{\mathsf{pk}(B)} & \{\!|v_0|\!\}_{\mathsf{hash}(N_a \,\hat{}\, N_b)} \\
\bullet \arrow[u] \arrow[r, Rightarrow] & \bullet \arrow[r, Rightarrow] & \bullet \arrow[u] \arrow[r, Rightarrow] & \bullet \arrow[u] \\
\bullet \arrow[r, Rightarrow] & \bullet \arrow[r, Rightarrow] \arrow[d] & \bullet \arrow[r, Rightarrow] & \bullet \\
\{\!|A \,\hat{}\, N_a|\!\}_{\mathsf{pk}(B)} \arrow[u] & \{\!|N_a \,\hat{}\, N_b \,\hat{}\, B|\!\}_{\mathsf{pk}(A)} & \{\!|N_b|\!\}_{\mathsf{pk}(B)} \arrow[u] & \{\!|x|\!\}_{\mathsf{hash}(N_a \,\hat{}\, N_b)} \arrow[u]
\end{tikzcd}$$

Figure 17: Π^i: NSL with encrypted message transport

We claim that in this case, the protocols Π^i_t, Π^r_t preserve the goals of Π^i and Π^r respectively. We also have some empirical evidence to support this: We have tested a variety of instances, and find that CPSA emits results that are exactly parallel for all of them.

This piggy-backing idea can also be extended to a sequence of messages, essentially justifying the properties of the j^{th} message by using a node index function g that maps the fourth message of Π^r to the j^{th} of the target protocol. We will not discuss that further here, however.

NSL key agreement, with message transport. Let Π^i be a variant of NSL in which there is an additional, final message. This message consists of a basic value v_0 of sort *data*, encrypted using a key created by hashing the two nonces for this session: $\mathsf{hash}(N_a \,\hat{}\, N_b)$. The initiator sends a message of the form $\{\!|v_0|\!\}_{\mathsf{hash}(N_a \,\hat{}\, N_b)}$, and the responder is willing to receive any message of the form $\{\!|x|\!\}_{\mathsf{hash}(N_a \,\hat{}\, N_b)}$ where x is a variable of sort message (Fig. 17). Π^r is the corresponding protocol in which the last message is transmitted by the responder and received by the initiator.

We have an obvious "inclusion" transformation $T^i \colon \mathsf{NSL} \to \Pi^i$. It maps the NSL initiator strand to the initiator strand of Π^i, and the responder strand to the responder strand. The node index functions are the identity, and the substitutions γ are also the identity function. We have a corresponding "inclusion" transformation $T^r \colon \mathsf{NSL} \to \Pi^r$.

For a set of starting points for NSL, including authentication claims for the two parties, and secrecy for the nonces N_a, N_b, we observe that the whole CPSA analysis uses only tests in which the critical values are always the nonces N_a, N_b. Since Π^i and Π^r provide no new message transmissions containing nonces as ingredients, it is immediate that the simulation condition of Thm. 3.9 is met. The liveness condition holds simply because the syntactic structure of the encryptions is unchanged.

This protocol enjoys the usual authentication and secrecy properties of NSL. In addition, it ensures secrecy for the data value $v_0 \in \mathsf{unique}$, assuming that the initiator has chosen the nonce $N_a \in \mathsf{unique}$, and the long term private keys $K_A^{-1}, K_B^{-1} \in \mathsf{non}$.

Piggybacking messages. We now define a family of protocols. Each one differs from Π^i only in the last message transmitted by the initiator, which may be replaced by any message that may contain arbitrary values of sort *data*, but has no nonces N as ingredients. For each message t such that there is no nonce $N \sqsubseteq t$, let Π^i_t be the result of replacing the initiator's last message by t.

The message t may contain several data values. Π^i_t provides confidential transport for them all by *piggybacking* them all on the message $\{\!|v_0|\!\}_{\mathsf{hash}(N_a \,\hat{}\, N_b)}$.

To prove this, consider a particular data value $d \sqsubseteq t$. Let the transformation $T_{d,t} \colon \Pi^i \to \Pi^i_t$ map the initiator role of Π^i to the initiator role of Π^i_t, using the identity function on node indices, and using the substitution $\gamma = [v_0 \mapsto d]$. It acts as the identity on the responder role. We can check that this is a transformation; in particular, Clause 4, which is the only one that sometimes requires care, is vacuously satisfied because the two different roles start with a transmission and a reception, and therefore have no common instances.

To define a function Δ on labels, map $\{\!|v_0|\!\}_{\mathsf{hash}(N_a \,\hat{}\, N_b)}$ to $\{\!|t|\!\}_{\mathsf{hash}(N_a \,\hat{}\, N_b)}$, and map all other occurrences of v_0 to d. We extend this to all other messages homomorphically. Thus, the critical value and escape set of a test are transformed by this function Δ.

For every PSLTS $\overset{\ell}{\rightsquigarrow}_1$ and Π^i-skeleton $\mathbb{A}$, $T(\mathbb{A})$ will be $\overset{\Delta(\ell)}{\rightsquigarrow}_2$-live if $\mathbb{A}$ is $\overset{\ell}{\rightsquigarrow}_1$-live.[3] For, $T(\mathbb{A})$ has the same nodes as $\mathbb{A}$ with the additional nested message structure t in place of v_0. This skeleton cannot be realized unless $\mathbb{A}$ is.

Indeed, no authentication test in $T(\mathbb{A})$ can have additional outcomes beyond the lifted results for $\mathbb{A}$. These tests always concern messages encrypted with K_A, K_B, or $\mathsf{hash}(N_a \,\hat{}\, N_b)$. None of these values can be disclosed, in any skeleton that respects our assumptions $K_A^{-1}, K_B^{-1} \in \mathsf{non}$, $N_a \in \mathsf{unique}$. Moreover, Π^i and Π^i_t produce messages with ingredient N_a in exactly the same way. Neither ever removes any ingredient from an encryption with key $\mathsf{hash}(N_a \,\hat{}\, N_b)$.

We have confirmed these claims by tests with CPSA. They show that with a variety of messages t containing no nonces as ingredients, CPSA carries out exactly the same sequence of tests as t varies. Textual diffs on its output confirm that it emits the same results for each of these tests, with only the expected changes in the message contents. The sizes and order of the generated cohorts agree. CPSA behaves differently if $N_a \sqsubseteq t$, although the confidentiality results appear to hold in this case also.

The protocol Π^r, in which the responder transmits $\{\!|v_0|\!\}_{\mathsf{hash}(N_a \,\hat{}\, N_b)}$ and the initiator receives $\{\!|x|\!\}_{\mathsf{hash}(N_a \,\hat{}\, N_b)}$, behaves very similarly. For each message t such that there is no nonce $N \sqsubseteq t$, we can form Π^r_t, and we have a similar family of transformations $T_{d,t} \colon \Pi^r \to \Pi^r_t$.[4]

[3] Figs. 11–12 show the authentication analysis LTS for NSL with secrecy information in Figs. 13–14, which may be useful for visualizing $\rightsquigarrow_1$.

[4] CPSA inputs and outputs for NSL, Π^i, Π^r, and three protocols of each of the forms Π^i_t and Π^r_t are available at `http://web.cs.wpi.edu/~guttman/transformations/tr_cpsa.tgz`. Diffs between the protocols Π^i and the sample Π^i_t show that all tests are chosen in exactly corresponding ways, and produce exactly corresponding cohorts. Likewise for Π^r.

4 Languages for Security Goals

We now consider languages of first order predicate logic. We introduce these languages to make our two main claims precise. One is that the Thm. 2.5 codifies a method (a possibly non-terminating algorithm) for determining whether a protocol meets security goals. The second is that Thm. 3.9 gives a criterion on protocol transformations, ensuring that security goals achieved by a source protocol will hold for the result of the transformation.

From a foundational point of view it is striking that the methods of the previous sections lead to preservation of truth for a broad class of formulas of a sharply well-defined syntactic form.

For this, we need languages in which to express the security goals for protocols. We introduce those languages in this section. The security goals are the formulas of the form

$$\forall \overline{x} \,.\, (\Phi \longrightarrow \Psi) \tag{2}$$

where the formulas Φ, Ψ may use $\wedge, \vee$, and $\exists$, but not $\neg$, $\longrightarrow$, or $\forall$. In fact, we focus on those in which Φ is "role-specific," a kind of typing constraint given in Def. 4.12, although this is a light condition. For each protocol Π, we provide a language for its security goals $\mathcal{GL}(\Pi)$. Each protocol transformation $T \colon \Pi_1 \to \Pi_2$ determines a translation from $\mathcal{GL}(\Pi_1)$ to $\mathcal{GL}(\Pi_2)$.

In designing these languages, we have adhered to three principles:

1. The vocabulary of $\mathcal{GL}(\Pi)$ should be derived directly from the roles and parameters of Π, and from the fragment structuring properties such as $\preceq$, uo, and non.

2. The formulas of $\mathcal{GL}(\Pi)$ should not describe the forms of the messages sent and received by the roles of Π, but should reflect only the parameters of the roles. This makes it easy for the formulas to be preserved under transformations, which may change the syntactic structures of the messages in use.

3. We will adhere to a pure first order logic, with no numbers or other data types that are extraneous to the protocols and their execution. This makes the metalogic of the languages $\mathcal{GL}(\Pi)$ very simple, with no unnecessary expressiveness due to these data types.

We start by showing that many security properties can in fact be expressed using our restricted means.

4.1 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. 2. 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. 2. 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 discuss the authentication goal from Fig. 8, p. 21. There is a similar goal starting from an initiator strand, which extends the step shown in Fig. 4. 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 for $\mathsf{pk}(A)^{-1}$—every execution exhibits at least the structure shown in the fragment on the right. More formally, every homomorphism $J \colon \mathcal{F} \mapsto \mathcal{D}$, where $\mathcal{D}$ is any *realized* fragment, factors through H. In particular, there exists some $K \colon \mathcal{E} \mapsto \mathcal{D}$ 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.

This security goal is not achieved by NS; indeed, Fig. 6 shows a counterexample (p. 6). The map that embeds $\mathcal{F}$ into Fig. 6 cannot factor through the H in Fig. 8. The initiator strand in Fig. 6 contains the parameter C for its peer, in contrast to the responder's parameter B. But in $\mathcal{E}_0$, we already have these two strands agreeing on the name of the responder. Hence, in every homomorphic image of $\mathcal{E}_0$, the two strands still agree on this parameter. Lowe's corrected protocol NSL [44] does achieve the goal, though.

We express this authentication goal as a formula by writing out the contents of the two fragments in Fig. 8 in formulas. The fragment $\mathcal{F}_0$ on the left 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 $\mathcal{F}_0$:

$$\begin{gathered}\texttt{RespThd}(n) \wedge \texttt{RespScd}(m) \wedge \texttt{Coll}(m,n) \wedge \\ \texttt{Peer}(m,a) \wedge \texttt{Self}(m,b) \wedge \texttt{MyNonce}(m,c) \wedge \texttt{YourNonce}(m,d) \wedge \\ \texttt{Non}(\texttt{inv}(\texttt{pk}(a))) \wedge \texttt{UnqAt}(m,c).\end{gathered} \tag{3}$$

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 $\mathcal{E}_0$ on the right, which also contains an initiator strand with matching parameters. We write this (incorporating Eqn. 3) in the form:

$$\begin{gathered}(3) \wedge \texttt{InitThd}(n') \wedge \\ \texttt{Self}(n',a) \wedge \texttt{Peer}(n',b) \wedge \texttt{YourNonce}(n',c) \wedge \texttt{MyNonce}(n',d)\end{gathered} \tag{4}$$

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

$$(3) \;\longrightarrow\; \exists n' \,.\, (4) \tag{5}$$

Observe that this formula is not valid in all fragments; for instance, its hypothesis is satisfied in $\mathcal{F}_0$, but its conclusion is not. However, can we make a *realized* fragment that satisfies Eqn. 3, but without also satisfying Eqn. 4?

If so, Π 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, p. 21. 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

$$(3) \;\longrightarrow\; \exists n', b' \,.\, (7) \tag{6}$$

where the conclusion uses the weakened formula:

$$\begin{gathered}(3) \wedge \texttt{InitThd}(n') \wedge \\ \texttt{Self}(n',a) \wedge \texttt{Peer}(n',b') \wedge \texttt{YourNonce}(n',c) \wedge \texttt{MyNonce}(n',d)\end{gathered} \tag{7}$$

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.

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 $\mathcal{F} \not\mapsto \mathcal{D}$.

As an example, Fig. 10, p. 22, 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 $\not\mapsto \cdot$, we mean to convey that there is no realized fragment $\mathcal{D}$ such that $\mathcal{F} \mapsto \mathcal{D}$.

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 $\mathcal{F} \not\mapsto \cdot$.

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 such as the Electronic Payment via Money Order protocol EPMO [40], 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 ℓ is a listener node, and $\mathtt{Hear}(\ell, c)$ to say that the value heard on ℓ is c, we may express the content of the fragment $\mathcal{F}$ in Fig. 10 (again incorporating Eqn. 3) in the form:

$$(3) \wedge \mathtt{Lsn}(\ell) \wedge \mathtt{Hear}(\ell, c) \wedge \mathtt{Non}(\mathtt{inv}(\mathtt{pk}(b))) \tag{8}$$

and the secrecy goal has this as its premise, and asserts that it cannot occur.

Its conclusion is the empty disjunction $\bot$.

$$(8) \longrightarrow \bot. \tag{9}$$

It is achieved in NSL but not NS. That is, Eqn. 8 is satisfied in no *realized* fragment for NSL.

4.2 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 Π, to express goals like those considered in Section 4.1. The diagrammatic goals are all expressed by implications between geometric formulas of $\mathcal{GL}(\Pi)$, i.e. by formulas of the form of Eqn. 2. 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. 2, we can use α-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:

$$\forall \overline{x} \,.\, (\phi \quad \longrightarrow \quad (\exists \overline{y_1} \,.\, \psi_1) \vee (\exists \overline{y_2} \,.\, \psi_2) \vee \ldots \vee (\exists \overline{y_j} \,.\, \psi_j)) \tag{10}$$

where: (i) ϕ and each ψ_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 (10).

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 [55], 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 Π'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.

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

The vocabulary of the language $\mathcal{GL}(\Pi)$ has two parts. One part is independent of Π, and is present in the language for every Π.

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 2.

The protocol-independent part also includes the predicate symbols shown in Table 2. $\mathtt{Preceq}(m, n)$ 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-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. $\mathcal{GL}(\mathsf{NS})$ 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 the parameter predicate $\mathtt{YourNonce}(n, v)$. There is no v such that this n and v satisfy $\mathtt{MyNonce}(n, v)$, 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 $\mathtt{MyNonce}(m, v)$.

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

Moreover, τ_p gives the parameter predicate for a particular role and parameter. For instance, in NS, $\tau_p(\mathsf{Init}, A)$ is $\mathtt{Self}$ and $\tau_p(\mathsf{Init}, N_a)$ is $\mathtt{MyNonce}$. However, $\tau_p(\mathsf{Resp}, A)$ is $\mathtt{Peer}$ and $\tau_p(\mathsf{Resp}, N_a)$ is $\mathtt{YourNonce}$. We do not assume that τ_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 ρ and ρ', then it has the same instance for a in ρ in the first case as for b in ρ' in the second. This makes the definition of satisfaction (Defn. 4.2) unambiguous. The tables τ_r, τ_p match roles and parameters of Π with the vocabulary of $\mathcal{GL}(\Pi)$. The tables for of $\mathcal{GL}(\mathsf{NS})$ are shown in Table 3.

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

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

τ_r	1	2	3
Init	`InitFst`	`InitScd`	`InitThd`
Resp	`RespFst`	`RespScd`	`RespThd`

τ_p	A	B	N_a	N_b
Init	`Self`	`Peer`	`MyNonce`	`YourNonce`
Resp	`Peer`	`Self`	`YourNonce`	`MyNonce`

Table 3: τ_r and τ_p for $\mathcal{GL}(\mathsf{NS})$

instances of ρ, 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 that strand behaves like a run of ρ so long as 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 ρ; i.e.* $\mathsf{match}(s, \rho, i) = \alpha$ *iff α 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 α, which are essentially homomorphisms from the message algebra 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 η is a map from variables to values which are either messages in the message algebra* 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 η 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.*[5]

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 4. Any atomic formula containing t is false if $\eta(t)$ is undefined.

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

[5]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 4: Clauses for satisfaction

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

Proof. Recall that the function symbols of $\mathcal{GL}\,\Pi$ contain only the key function symbols $\texttt{pk}(a)$, $\texttt{sk}(a)$, and $\texttt{inv}(k)$.

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

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

If $\mathcal{F},\,\eta \models \texttt{ParamPred}(t,t')$, then there is a ρ 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 Φ,Ψ is preserved under H, then so is satisfaction for $\Phi \wedge \Psi$, $\Phi \vee \Psi$, and $\exists x\,.\,\Phi$. $\square$

Φ *entails* Ψ if for all Π-fragments $\mathcal{F}$, $\mathcal{F},\eta \models \Phi$ implies $\mathcal{F},\eta \models \Psi$. Φ and Ψ are *equivalent* if each entails the other. Two variable assignments η,θ *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 Φ, and write $\eta \sim_{\Phi} \theta$ to mean $\eta \sim_{\mathsf{fv}(\Phi)} \theta$.

Definition 4.4 *Let Π be a protocol, and let $G \in \mathcal{GL}(\Pi)$ be a closed formula of the form (10). Π* achieves *the goal G if, for every* realized *fragment $\mathcal{D}$, $\mathcal{D} \models G$.*

4.3 Characteristic Fragments and Formulas

Each of the fragments in Figs. 8–10 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 3.

This suggests the idea of a *characteristic formula*: A conjunction of atomic formulas Φ is a characteristic formula for $\mathcal{F}$ if Φ 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 4.5 *Suppose $\mathcal{F}$ is a Π-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 η 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 η. Let D be the set of atomic formulas ϕ 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 η-diagram of $\mathcal{F}$, *written $\delta_\eta(\mathcal{F})$, is the conjunction $\bigwedge D$.*

Whenever we write $\delta_\eta(\mathcal{F})$, we assume that η is injective and covers $S = \mathsf{nodes}(\mathcal{F}) \cup \mathsf{Params}(\mathcal{F})$. When η 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 ψ is a consequence of $\delta(\mathcal{F})$, as long as we specify the meanings of any additional variables in ψ:

Lemma 4.6 *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 ψ.*

Proof. If $\mathcal{F}, \theta \models \psi$, where ψ 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 η, say $\eta(x)$, so use the equation $x = y$. In the last case, a is in the range of η, say $\eta(x)$, so use the equation $f(x) = y$. □

A characteristic formula is any formula equivalent to the diagram:

Definition 4.7 *A conjunction of atomic formulas Φ is a* characteristic formula for $\mathcal{F}$ via η, *written $\Phi \in \mathsf{cform}_\eta(\mathcal{F})$, if Φ is equivalent to $\delta_\eta(\mathcal{F})$.*

As examples, each of the formulas we mentioned in Section 4.1 as giving the "content" of one of our fragments 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 4.8 *If $\Phi, \Psi \in \mathsf{cform}_\eta(\mathcal{E})$, then Φ and Ψ 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. 3 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. 3, by Lemma 4.3. Indeed, conversely, any fragment that satisfies Eqn. 3 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. 3. 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. 3, because being a homomorphic image of $\mathcal{F}$ characterizes whether a fragment satisfies Eqn. 3.

Definition 4.9 *$\mathcal{E}$ is a* characteristic fragment *for Φ* via *variable assignment η iff for every $\mathcal{F}$ and assignment θ,*

$$\mathcal{F}, \theta \models \Phi \text{ iff there exists a unique } H\colon \mathcal{E} \mapsto \mathcal{F} \text{ such that } (H \circ \eta) \sim_\Phi \theta. \tag{11}$$

Lemma 4.10 *If $\mathcal{E}$ and $\mathcal{F}$ are characteristic fragments for Φ via η and θ resp., then there is an isomorphism $H\colon \mathcal{E} \mapsto \mathcal{F}$ 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 $K\colon \mathcal{F} \mapsto \mathcal{E}$. Since there is only one homomorphisms $\mathcal{E} \mapsto \mathcal{E}$ satisfying Eqn. 11, and the identity is one, $K \circ H$ must also be the identity. So H is an isomorphism. □

Because of Lemma 4.10, 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 $\mathcal{F}$ is a characteristic fragment for a formula Φ, and Φ' is a characteristic formula for $\mathcal{F}$, then Φ and Φ' are equivalent. When Φ is a characteristic formula for $\mathcal{F}$, and $\mathcal{E}$ is a characteristic fragment for Φ, then $\mathcal{E} \mapsto \mathcal{F}$.[6]

Taking a characteristic formula may discard some information, because $\mathcal{GL}(\Pi)$ is of limited expressiveness, but taking a characteristic fragment does not.

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

$\mathsf{cfrag}(\mathsf{cform}(\mathcal{F}))$ may be properly less informative than $\mathcal{F}$, in the sense that the homomorphism $\mathsf{cfrag}(\mathsf{cform}(\mathcal{F})) \to \mathcal{F}$ is not an isomorphism.

To take the simplest case, suppose that $\mathcal{F}$ contains only $\bullet \overset{\{\!|a|\!\}_K}{\leftarrow}$. 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 $\bullet \overset{x}{\leftarrow}$, where x is an indeterminate that can be replaced by any message. In particular, $\alpha = x \mapsto \{\!|a|\!\}_K$ determines the non-isomorphism from $\bullet \overset{x}{\leftarrow}$ to $\bullet \overset{\{\!|a|\!\}_K}{\leftarrow}$.

Lemma 4.11 *1. If $\Phi \in \mathsf{cform}_\eta(\mathcal{F})$ and $\mathsf{cfrag}_\theta(\Phi) = \mathcal{E}$, then $\mathcal{E} \to \mathcal{F}$.*

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

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 $H : \mathcal{E} \to \mathcal{F}$ exists, where $\eta \sim_\Phi (H \circ \theta)$.

2. (a) Ψ entails Φ, because the latter is a conjunction of atomic formulas all satisfied in $\mathcal{F}$, and Lemma 4.6 ensures that $\delta_\eta(\mathcal{F})$ entails each of them.

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

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 4.12 *Let Φ be a conjunction of atomic formulas.*

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

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

Φ is role-specific *iff*

1. *its node variables and message variables partition $\mathsf{fv}(\Phi)$;*

2. *only message variables appear as argument to a key function, $\mathtt{Unq}$, $\mathtt{Non}$, or in the second position of $\mathtt{UnqAt}$; and*

3. *only node variables appear (a) as arguments to $\mathtt{Preceq}$ or $\mathtt{Coll}$, or (b) as the first argument to $\mathtt{UnqAt}$ or a parameter predicate.*

All of the characteristic formulas in Section 4.1 are role-specific.

Lemma 4.13 *Let Φ be role-specific, with $\mathcal{F}, \eta \models \Phi$. If x is a node variable of Φ, then $\eta(x) \in \mathsf{nodes}(\mathcal{F})$. If x is a message variable of Φ, 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 η in Defn. 4.5, this partitions the variables in $\delta_\eta(\mathbb{A})$. The remaining conditions follow from the type constraints in the definition of satisfaction. $\square$

$\delta_\eta(\mathcal{F})$ 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 4.14 *Let Φ 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 $\mathsf{skeleton}(\mathcal{F}) \mapsto \mathcal{F}$.

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 $\mathsf{skeleton}(\mathcal{F})$. $\square$

We now prove that $\mathsf{cfrag}(\Phi)$ is well-defined for role-specific Φ. 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 4.15 *Suppose that $H \colon \mathcal{F} \mapsto \mathcal{E}$.*

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

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

Theorem 4.16 *If a satisfiable conjunction Φ of atomic formulas is role-specific, then $\mathsf{cfrag}_\eta(\Phi)$ is well-defined, and is a skeleton, for some η.*

Proof. We will assume that Φ 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, Φ 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 $\mathsf{cfrag}(\Phi)$. Indeed, it is a skeleton. The variable assignment η 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 4.14, $\mathsf{skeleton}(\mathcal{F}), \theta \models \Phi \wedge \phi_j$. Since $\mathsf{cfrag}_\eta(\Phi) = \mathbb{A}$, there is a (unique) H such that $H\colon \mathbb{A} \rightarrow \mathsf{skeleton}(\mathcal{F})$, and $\theta \sim_\Phi H \circ \eta$.

Since we have H, we may apply the clauses of Lemma 4.15 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 ϕ_j:

ϕ_j is $\mathtt{Preceq}(m, n)$: 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.

ϕ_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 $s' \downarrow k$.

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

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

ϕ_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.

ϕ_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.

ϕ_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 ρ 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 η', we map $n \mapsto (s \downarrow i)$, and for $\mathsf{fv}(\Phi)$, η' agrees with η.

ϕ_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 ρ, 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 η' be η 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 η' be η 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 θ for the formula. □

By Lemma 4.13 and Thm. 4.16, we can always assume that $\mathsf{cfrag}(\Phi)$ is defined when $\Phi \in \mathsf{cform}(\mathbb{A})$. Even if Φ 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. 1.7 and 4.4).

Our assumption in this theorem that each ψ_i entails ϕ is not a significant restriction. Since ϕ is already available as a hypothesis in the goal formula, we could replace any ψ_i that did not meet this assumption by $\phi \wedge \psi_i$ without changing the meaning of the formula. Indeed, in case ψ_i is a characteristic formula, it is already of this form, as we illustrated in Section 4.1.

Theorem 4.17 *Let* Π *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 ϕ, ψ_i *are satisfiable and role-specific, and each* ψ_i *entails* ϕ*. Then there exists a family of homomorphisms*

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

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

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

Proof. By Thm. 4.16, $\mathsf{cfrag}_\eta(\phi)$ and each $\mathsf{cfrag}_{\theta_i}(\psi_i)$ is a well-defined skeleton. Since ψ_i entails ϕ, the latter is satisfied in $\mathsf{cfrag}_{\theta_i}(\psi_i)$. Since $\mathsf{cfrag}_\eta(\phi)$ is a characteristic fragment, there is a homomorphism $H_i \colon \mathsf{cfrag}_\eta(\phi) \mapsto \mathsf{cfrag}_{\theta_i}(\psi_i)$ such that $\theta_i \sim_\phi H_i \circ \eta$. We use the family $\{H_i\}_{1 \le i \le j}$ of these homomorphisms.

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

   If $K \colon \mathsf{cfrag}_\eta(\phi) \mapsto \mathcal{D}$, 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 ζ 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 $J \colon \mathsf{cfrag}_{\theta_i}(\psi_i) \mapsto \mathcal{D}$ 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 ζ 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 $K \colon \mathsf{cfrag}_\eta(\phi) \mapsto \mathcal{D}$ 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 $\mathcal{D}, \zeta \models \exists \overline{y_i} \,.\, \psi_i$. □

Combining this with Thm. 2.5, we have:

Corollary 4.18 *Let ϕ be role specific, and let G be $\forall \overline{x} \,.\, (\phi \longrightarrow \bigvee_i \exists \overline{y_i} \,.\, \psi_i)$. Let $\cdot \overset{\cdot}{\rightsquigarrow} \cdot$ be a* PSLTS*, and $\mathsf{cfrag}_\eta(\phi) \in S(\rightsquigarrow)$. Then Π achieves G iff, for every sufficiently long path*

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

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. 4.17 to the (possibly infinite) family $\{H_{\sigma_\mathcal{D}}\}_\mathcal{D}$. □

Another corollary, due originally to Ramsdell [55, Thm. 2], works in the opposite direction, constructing a formula from a family that Π enforces:

Corollary 4.19 *Let $\{H_i \colon \mathbb{A} \mapsto \mathcal{E}_i\}_{1 \le i \le j}$ 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 Π enforces $\{H_i\}_{1 \le i \le j}$, then Π achieves goal formula

$$\phi \longrightarrow \bigvee_{1 \le i \le j} \exists \overline{y_i} \,.\, \psi_i. \tag{12}$$

Proof. Applying Thm. 4.17 to Formula 12 as G, we generate a family

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

Thm. 4.17 tells us that it suffices for us to show that Π enforces $\{K_i\}_{1 \le i \le j}$. By Lemma 4.11, for each i there is an L_i such that $L_i \colon \mathsf{cfrag}_{\theta_i}(\psi_i) \mapsto \mathcal{E}_i$. 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 $M \colon \mathbb{A} \mapsto \mathcal{D}$. Since Π enforces $\{H_i\}_{1 \le i \le j}$, 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 Π enforces $\{K_i\}_{1 \le i \le j}$. □

In Thm. 4.17 and Cor. 4.19, the finiteness of the index set $I = \{i \colon 1 \le i \le j\}$ 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.

4.4 Preserving Goal Formulas

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

By the definition of *role specific*, Defn. 4.12, if a variable v occurs in ϕ, 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 (ρ, i) if in the table τ_r for $\mathcal{GL}(\Pi)$, the entry for (ρ, i) is R. Given ϕ and a node variable n in ϕ, we will say that its *role position location in* ϕ is (ρ, i) if the leftmost role position predicate in ϕ containing n is $R(n)$, and the role position location of R is (ρ, 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* (ρ, a) in ϕ if:

- n has role position location (ρ, i) in ϕ, for some i; and
- in the table τ_p for $\mathcal{GL}(\Pi)$, the entry for (ρ, 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 4.20 *Let $T\colon \Pi_1 \to \Pi_2$. The T-translation of a role specific formula ϕ is the result of replacing each predicate symbol R within it according to the following rules:*

- *If R belongs to the protocol-independent vocabulary $=$, `Preceq`, `Coll`, `Unq`, `UnqAt`(n, v), `Non`(v), then it is unchanged.*
- *Suppose R is a role position predicate, and the role position location of R is (ρ_1, i), and let $T(\rho_1) = \rho_2, g, \gamma$.*

  Replace R with R', which is the entry in τ_r for $\mathcal{GL}(\Pi_2)$ for ρ_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 (ρ, a) in ϕ, and let $T(\rho_1) = \rho_2, g, \gamma$.*

  Replace R with R', which is the entry in τ_p for $\mathcal{GL}(\Pi_2)$ for ρ_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 Γ with role specific ϕ, 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 ϕ 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 ψ_i are not role specific.

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

Definition 4.21 *Let $T\colon \Pi_1 \to \Pi_2$ and $\mathbb{A} \in \mathsf{Skel}(\Pi_1)$, and let η be a variable assignment $\eta\colon \mathsf{Var} \to \mathsf{nodes}(\mathbb{A}) \cup \mathsf{ALG}$. Let lift_T lift $\mathsf{nodes}(\mathbb{A})$ to $\mathsf{nodes}(T(\mathbb{A}))$.*

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

Lemma 4.22 *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. 4.16 applies.

2. By induction on the structure of the conjunction ϕ. Essentially, one checks that Defn. 4.20 matches Defn. 3.4.

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(η), whence every strand in $T(\mathbb{A})$ has a node in range$(\overline{\eta})$. For α, 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. 4.9 entails that there is a $K\colon \mathsf{cfrag}(T(\phi)) \to T(\mathsf{cfrag}(\phi))$. Hence, by the uniqueness in Def. 4.9, $J \circ K = \mathsf{Id}$. Hence, $T(\mathsf{cfrag}(\phi))$ and $\mathsf{cfrag}(T(\phi))$ are isomorphic. □

We now turn to our last main theorem.

Theorem 4.23 (Goal Preservation) *Let ϕ 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* PSLTSs *for Π_1 and Π_2 resp., with $\mathsf{cfrag}(\phi) \in S(\rightsquigarrow_1)$ and $T(\mathsf{cfrag}(\phi)) \in S(\rightsquigarrow_2)$. Suppose*

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

Then for every security goal $\Gamma = \forall \vec{x} \,.\, \phi \longrightarrow \bigvee_i \exists \vec{y_i} \,.\, \psi_i$, if Π_1 achieves Γ, then Π_2 achieves $T(\Gamma)$.

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

Suppose that $\mathbb{C}$ is any realized Π_2-skeleton, and θ is a variable assignment into $\mathsf{nodes}(\mathbb{C}) \cup \mathsf{ALG}$. If $\mathbb{C}, \theta \not\models T(\phi)$, then $\mathbb{C}, \theta \models T(\phi) \;\longrightarrow\; \bigvee_i \exists \vec{y_i} \,.\, T(\psi_i)$.

So suppose $\mathbb{C}, \theta \models T(\phi)$. By Defn. 4.9, there is an $H \colon \mathsf{cfrag}_{\overline{\eta}}(T(\phi)) \mapsto \mathbb{C}$, and $\theta \sim_\phi H \circ \overline{\eta}$. Moreover,

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

by Lemma 4.22, Clause 4. So we may apply Thm. 3.9: There is a Π_1-realized $\mathbb{B}$ such that $\mathsf{cfrag}_\eta(\phi) \overset{\sigma}{\rightsquigarrow}{}_1^* \mathbb{B}$, and, for $K = T(H_\sigma)$,

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

If Π_1 achieves Γ, then for some i, ψ_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 $J \circ \overline{\zeta} \sim_{T(\phi)} (H \circ \overline{\eta})$. □

Continuing the examples Π^i, Π^r of Section 3.4, the inclusion transformations $T^i \colon \mathsf{NSL} \to \Pi^i$ and $T^r \colon \mathsf{NSL} \to \Pi^r$ immediately ensures that the authentication and secrecy formulas that CPSA establishes for NSL hold for Π^i, Π^r. On the same assumptions, together with $\mathtt{Non}(v)$, where v is the data payload parameter, we can check that v remains secret. I.e. if $\mathtt{Lsn}(\ell) \wedge \mathtt{Hear}(\ell, v)$, then $\perp$ follows.

With this result in hand, Thm 4.23 now entails that the same formula holds of any of the piggybacked data values in any of the family of protocols Π^i_t and Π^r_t. This wholesale justification of a class of secrecy properties is a new contribution of this work.

Future work. We leave a major gap: What syntactic property of $T \colon \Pi_1 \to \Pi_2$ ensures that T preserves security goals? A clue comes from the "disjoint encryption" property [38, 33], cf. [47, 14]. Consider a map E from all encrypted units used by Π_1 to a subset of the encrypted units of Π_2. Π_2 should create an encryption $\alpha(E(e))$ on node n only if $n = F(n_0)$ and n_0 creates $\alpha(e)$ in Π_1. Likewise, Π_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 Π_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, and an apology. Thanks to Dan Dougherty, Dusko Pavlovic, John Ramsdell, Gabrielle Raymond, Paul Rowe, and Javier Thayer. I presented early versions of this material 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.

The apology is due to Robert Lowell, who wrote, "Yet why not say what happened?" My version emphasizes our desire to find the generic range of possibilities rather than—his concern—the specifics of a recalled lived experience.

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. Technical Report 2000/067, IACR, October 2001. Appeared in 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 *Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, LNCS. Springer, December 2007.

[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.

[40] Joshua D. Guttman, F. Javier Thayer, Jay A. Carlson, Jonathan C. Herzog, John D. Ramsdell, and Brian T. Sniffen. Trust management in strand spaces: A rely-guarantee method. In David Schmidt, editor, *Programming Languages and Systems: 13th European Symposium on Programming*, number 2986 in LNCS, pages 325–339. Springer, 2004.

[41] Mei Lin Hui and Gavin Lowe. Fault-preserving simplifying transformations for security protocols. *Journal of Computer Security*, 9(1/2):3–46, 2001.

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

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

[44] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceeedings of* TACAS, volume 1055 of *LNCS*, pages 147–166, 1996.

[45] 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.

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

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

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

[49] 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.

[50] Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *CCS*, pages 166–175. ACM, 2001.

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

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

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

[54] 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.

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

[56] John D. Ramsdell and Joshua D. Guttman. CPSA: A cryptographic protocol shapes analyzer, 2009. `http://hackage.haskell.org/package/cpsa`.

[57] John D. Ramsdell, Joshua D. Guttman, Moses D. Liskov, 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.

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

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

[60] 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.

[61] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Mixed strand spaces. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*. IEEE CS Press, June 1999.

[62] 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.