Security Goals and Protocol Transformations*

Joshua D. Guttman

Worcester Polytechnic Institute

Abstract. Cryptographic protocol designers work incrementally. Having achieved some goals for confidentiality and authentication in a protocol Π_1, they transform it to a richer Π_2 to achieve new goals.
But do the original goals still hold? More precisely, if a goal formula Γ holds whenever Π_1 runs against an adversary, does a translation of Γ hold whenever Π_2 runs against it?
We prove that a transformation preserves goal formulas if a labeled transition system for analyzing Π_1 simulates a portion of an LTS for analyzing Π_2, while preserving progress in that portion.
Thus, we examine the process of analyzing a protocol Π. We use LTSs that describe *our* activity when *analyzing* Π, not that of the principals *executing* Π. Each analysis step considers—for an observed message reception—what earlier transmissions would explain it. The LTS then contains a transition from a fragmentary execution containing the reception to a richer one containing an explaining transmission. The strand space protocol analysis tool CPSA generates some of the LTSs used.

1 Introduction

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

Our goal here is not to make this cleverness unnecessary, but to explain it semantically. Thm. 2 says how to show that a transformed protocol satisfies some security goals, when the source protocol did. Although a logical result about models of protocol behavior and the formulas they satisfy, it is a corollary of a logic-free theorem (Thm. 1). The latter concerns only fragments of protocol executions (called *skeletons*), the information-preserving maps (*homomorphisms*) between them, and some labeled transition systems. These LTSs formalize the

* Supported by the National Science Foundation under grant CNS-0952287.

activity of protocol analysis. Reifying protocol analysis into LTSs, and explaining relations between protocols using them, appear to be new in this paper.

Structure of this paper. Section 2 introduces two protocols, with two transformations between them, motivating Def. 1. Section 3 analyzes these protocols, illustrating how a transformation can preserve the activity of protocol analysis. Section 4 axiomatizes these analysis activities, representing them as labeled transition systems. A simulation-plus-progress relation on LTSs ensures that a transformation does not create counterexamples to security goals (Section 5).

Section 6 defines classical first order languages $\mathcal{L}(\Pi)$, and defines security goal translations. We lift Thm. 1 to satisfaction of goals (Thm. 2) in Section 7, and comment on related and future work. Elsewhere, we will propose syntactic conditions on message formats for goal preservation, easing use of our method.

Strand Spaces. We work within the strand space theory [17]. A *strand* is the sequence of transmissions and receptions executed by a single principal in a single protocol session. We will write strands, either horizontally or vertically, as sequences of bullets connected by double arrows: $\bullet \Rightarrow \bullet$.

A protocol Π consists of a finite set of strands, called the *roles* of the protocol, possibly annotated with some additional trust assumptions that we will not need here. A strand that is an *instance* of a role is a possible behavior of a principal complying with Π. These instances result from roles by filling in their parameters with values from some reasonable algebra of messages. Each of these strands is a *regular* behavior of some principal, i.e. a local session in which the principal complies with Π. Transmission and reception events jointly are *nodes*.

Executions (or fragmentary executions) consist of a number of regular strands or their initial segments. We call them *skeletons*. A skeleton $\mathbb{A}$ consists of its regular nodes, equipped with (i) a partial ordering $\preceq_{\mathbb{A}}$, related to the Lamport causal ordering [22]; (ii) some assumptions $\mathsf{unique}(\mathbb{A})$ about freshly chosen values; and (iii) some assumptions $\mathsf{non}(\mathbb{A})$ about uncompromised long term keys.

If n is a node receiving t in $\mathbb{A}$, an adversary may use transmissions prior to n in the partial ordering $\preceq_{\mathbb{A}}$ to derive t. A skeleton $\mathbb{A}$ is *realized* if, whenever n is a reception node in $\mathbb{A}$, an adversary can obtain or construct its message t, without violating the assumptions $\mathsf{unique}(\mathbb{A})$ and $\mathsf{non}(\mathbb{A})$.

2 Some Protocol Transformations

HD, one of the simplest possible authentication protocols, is a half-duplex, authentication-only subprotocol of Needham-Schroeder [24]. The Yes-or-No Protocol YN allows a Questioner to ask a question, to which the Answerer gives a private, authenticated reply; YN is constructed by two transformations of HD.

The Protocol HD. HD, as shown in Fig. 1, gives the initiator an authentication guarantee that the responder has participated; it gives the responder no guarantee.[1] No shared secret is established. The top half of the figure

[1] We write $t_0 \,\hat{}\, t_1$ for the concatenation of two messages, and $\{\!|t|\!\}_{\mathsf{pk}(B)}$ for the (asymmetric) encryption of t using the public encryption key of B.

shows the initiator role, first transmitting the encrypted message for B and then receiving the freed nonce N. The bottom half shows the responder role, first receiving the encrypted message and then freeing and transmitting N'. A regular strand of HD is any instance of either one of these roles, using any values for the parameters N, B, N', B'. A skeleton $\mathbb{A}$ over HD contains any number of regular strands. $\mathbb{A}$ is realized if an adversary can synthesize the message received on each reception node, using messages transmitted earlier relative to $\preceq_{\mathbb{A}}$, without violating the freshness and non-compromise assumptions $\mathsf{unique}(\mathbb{A})$ and $\mathsf{non}(\mathbb{A})$.

$$\begin{tikzcd}
\text{Init} & \bullet \arrow[r, Rightarrow] \arrow[d] & \bullet \\
& \{\!|N|\!\}_{\mathsf{pk}(B)} & N \arrow[u] \\
& \{\!|N'|\!\}_{\mathsf{pk}(B')} \arrow[d] & N' \\
\text{Resp} & \bullet \arrow[r, Rightarrow] & \bullet \arrow[u]
\end{tikzcd}$$

Fig. 1. One-sided authentication protocol HD

The Yes-or-No Protocol. In the Yes-or-No protocol, a Questioner asks a yes-or-no question, and an Answerer provides the answer. 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 to indicate a *yes*, and the second to indicate a *no*. No adversary learns anything, since whichever nonce was released, the questioner was equally likely to have used it in the other position.

The protocol has four roles (Fig. 2). 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.

$$\begin{tikzcd}
\text{QAf} & \bullet \arrow[r, Rightarrow] \arrow[d] & \bullet & \bullet \arrow[r, Rightarrow] \arrow[d] & \bullet & \text{QNg} \\
& \{\!|Q_1 \,\hat{}\, Y_1 \,\hat{}\, N_1|\!\}_{\mathsf{pk}(B_1)} & Y_1 \arrow[u] & \{\!|Q_2 \,\hat{}\, Y_2 \,\hat{}\, N_2|\!\}_{\mathsf{pk}(B_2)} & N_2 \arrow[u] & \\
& \{\!|Q_3 \,\hat{}\, Y_3 \,\hat{}\, N_3|\!\}_{\mathsf{pk}(B_3)} \arrow[d] & Y_3 & \{\!|Q_4 \,\hat{}\, Y_4 \,\hat{}\, N_4|\!\}_{\mathsf{pk}(B_4)} \arrow[d] & N_4 & \\
\text{AnAf} & \bullet \arrow[r, Rightarrow] & \bullet \arrow[u] & \bullet \arrow[r, Rightarrow] & \bullet \arrow[u] & \text{AnNg}
\end{tikzcd}$$

Fig. 2. The Yes-or-No Protocol YN

We can view the left half of this diagram as a transformation of the protocol HD, if we first rename the nonces N, N' of HD to the affirmative nonces Y_1, Y_3. Alternatively, if we instead rename the nonces N, N' to the negative nonces N_2, N_4, then we can view the right half of the diagram as a transformation of

HD. Thus, before formalizing the transformations, we formalize these renamings. The renamings, yielding respectively the protocols $\alpha_1(\mathsf{HD})$ and $\alpha_2(\mathsf{HD})$, are:

$$\alpha_1 = [N \mapsto Y_1,\ N' \mapsto Y_3,\ B \mapsto B_1,\ B' \mapsto B_3] \quad \text{and}$$
$$\alpha_2 = [N \mapsto N_2,\ N' \mapsto N_4,\ B \mapsto B_2,\ B' \mapsto B_4]$$

Protocol Transformations. By a *protocol transformation* from a source protocol Π_1 to a target protocol Π_2, we mean a map from nodes on roles of Π_1 to nodes on roles of Π_2. The images of nodes of a single role $\rho_1 \in \Pi_1$ all lie along a single role $\rho_2 \in \Pi_2$, so we formalize this with two components: one which selects the correct ρ_2, and another which is a function g that maps the index of a node on ρ_1 to the index of its image along ρ_2. Thus:

$F_1 \colon \alpha_1(\mathsf{HD}) \to \mathsf{YN}$ sends $\mathrm{Init}_{\alpha_1(\mathsf{HD})}$ to the Questioner's Affirmative role $\mathrm{QAf}_{\mathsf{YN}}$. The first node of $\mathrm{Init}_{\alpha_1(\mathsf{HD})}$—the transmission node—is associated with the first node of $\mathrm{QAf}_{\mathsf{YN}}$, and the second nodes are associated. F_1 sends $\mathrm{Resp}_{\alpha_1(\mathsf{HD})}$ to the Answerer's Affirmative role $\mathrm{AnAf}_{\mathsf{YN}}$, preserving node indices.
Thus, $F_1(\mathrm{Init}_{\alpha_1(\mathsf{HD})}) = (\mathrm{QAf}_{\mathsf{YN}}, \mathsf{Id})$, where Id is the identify function $\lambda i\,.\,i$. $F_1(\mathrm{Resp}_{\alpha_1(\mathsf{HD})}) = (\mathrm{AnAf}_{\mathsf{YN}}, \mathsf{Id})$.

$F_2 \colon \alpha_2(\mathsf{HD}) \to \mathsf{YN}$ acts similarly, with negative target roles. It sends $\mathrm{Init}_{\alpha_2(\mathsf{HD})}$ to the Questioner's Negative role $\mathrm{QNg}_{\mathsf{YN}}$. It sends $\mathrm{Resp}_{\alpha_2(\mathsf{HD})}$ to the Answerer's Negative role $\mathrm{AnNg}_{\mathsf{YN}}$. In both, F_2 preserves node indices.
Thus, $F_2(\mathrm{Init}_{\alpha_2(\mathsf{HD})}) = (\mathrm{QNg}_{\mathsf{YN}}, \mathsf{Id})$, and $F_2(\mathrm{Resp}_{\alpha_2(\mathsf{HD})}) = (\mathrm{AnNg}_{\mathsf{YN}}, \mathsf{Id})$.

These node index functions g are the identity, but other transformations use non-identity gs. For instance, if one principal in YN sent a message before the messages shown, which the other received before the messages shown, then we would alter F_1, F_2 to use $\lambda i\,.\,i+1$ to increment each node index. (See App. A.1.)

Terminology. We write $\rho \downarrow i$ to mean the i^{th} node along ρ, starting from 1. We write $t_0 \sqsubseteq t_1$ to mean that message t_0 is an *ingredient* in t_1, meaning that t_0 is a subterm of t_1 considering plaintexts but not the keys used to prepare encryptions. That is, $\sqsubseteq$ is the smallest reflexive transitive relation such that $t_0 \sqsubseteq \{\!|t_0|\!\}_K$; $t_0 \sqsubseteq t_0 \,\hat{}\, t_1$, and $t_1 \sqsubseteq t_0 \,\hat{}\, t_1$. The key K used in an encryption $\{\!|t_0|\!\}_K$ is not an ingredient of it, however, unless it was an ingredient of t_0 (contrary to good practice). For instance, $N_b \sqsubseteq \{\!|N_b \,\hat{}\, B|\!\}_{\mathsf{pk}(A)}$, but $\mathsf{pk}(A) \not\sqsubseteq \{\!|N_b \,\hat{}\, B|\!\}_{\mathsf{pk}(A)}$.

A message t_0 *originates at* a node n iff n is a transmission node, $t_0 \sqsubseteq \mathsf{msg}(n)$, and for all m such that $m \Rightarrow^+ n$, $t_0 \not\sqsubseteq \mathsf{msg}(m)$. Thus, t_0 originates when it was transmitted as an ingredient, but was neither transmitted nor received earlier on the same strand. △

Our examples have several properties. The node index mapping function $\lambda i\,.\,i$ is order-preserving, and the transformations also preserve the direction of the nodes (transmission vs. reception). The transmission node n of the HD initiator originates the value N and this value—as renamed by α_1—also originates on $F_1(n)$. The reception node m of the HD responder receives $N \sqsubseteq \mathsf{msg}(m)$, and we also have $\alpha_1(N) \sqsubseteq \mathsf{msg}(F(m))$. Thus, F_1 preserves the originated values and the ingredients of nodes. Similarly, F_2 preserves these properties of $\alpha_2(N)$.

These properties, with one other, define a protocol transformation. This last property is vacuously true of F_1, F_2. It concerns branching, as for instance, in a transformation $F\colon \mathsf{YN} \to \Pi$, QAf and QNg branch after a first node in common. It says that the result in the target Π should not commit to either branch until the source behaviors have committed by diverging from each other.

Definition 1 (Transformation). *Suppose F maps each role $\rho_1 \in \Pi_1$ to a pair ρ_2, g, where $\rho_2 \in \Pi_2$ and $g\colon \mathbb{N}^+ \to \mathbb{N}^+$. F is a* protocol transformation *iff:*

1. *g is order-preserving and $\rho_2 \downarrow g(\mathsf{length}(\rho_1))$ is well-defined;*
2. *$\rho_1 \downarrow i$ is a transmission (or resp. reception) node iff $\rho_2 \downarrow g(i)$ is;*
3. *Whenever $x \sqsubseteq \mathsf{msg}(\rho_1 \downarrow i)$, there exists a $j \leq g(i)$ such that $x \sqsubseteq \mathsf{msg}(\rho_2 \downarrow j)$;*
4. *Whenever x originates on $\rho_1 \downarrow i$, for some $j \leq g(i)$, x originates on $\rho_2 \downarrow j$;*
5. *Each common instance of both ρ_1 and $\sigma_1 \in \Pi_1$ up to node i yields a common instance of $F(\rho_1), F(\sigma_1)$ up to $g(i)$.*
   That is, assume $F(\sigma_1) = \sigma_2, h$ and $\alpha(\rho_1) \downarrow j = \beta(\sigma_1) \downarrow j$ for all $j \leq i$. Then $g(j) = h(j)$ for all $j \leq i$. Also, there is a β' that agrees with β on the parameters of σ_1 such that $\alpha(\rho_2) \downarrow j = \beta'(\sigma_2) \downarrow j$ for all $j \leq g(i)$.

3 Security Analysis of HD and YN

Security analysis aims to find what must have happened—or must not have happened—if a certain situation has arisen. In the case of HD, the relevant analysis considers what must have happened, when there has been a local session of the initiator role. If its nonce N was freshly chosen, and B's private decryption key was uncompromised, what can we be sure has happened?

CPSA [26] is a software tool to answer such questions. It starts with an object representing the situation—namely a skeleton $\mathbb{A}$ (cf. p. 2)—and generates enriched skeletons to represent all the complete executions compatible with the starting point $\mathbb{A}$. In each step it takes, CPSA locates an *unsolved test*, some reception node that cannot be explained by adversary activity, given the regular (non-adversarial) activity currently present in the skeleton. For each alternate piece of regular activity that could help explain the current skeleton, CPSA constructs an enriched skeleton; the search branches to explore those enrichments. When every reception node is explained, and the skeleton is realized, CPSA has found a leaf in the search ("a shape").

CPSA implements a labeled transition system. The nodes are skeletons. There is a transition $\mathbb{A} \overset{\ell}{\rightsquigarrow} \mathbb{B}_i$ if $\mathbb{B}_i$ is one of the alternate enrichments that explains a test ℓ unsolved in $\mathbb{A}$. All of the $\mathbb{B}_i$ that provide alternate solutions to ℓ are successors of $\mathbb{A}$ with the same label ℓ. Given a protocol Π, security analysis for authentication and confidentiality goals involving the situation $\mathbb{A}_0$ consists of exploring the portion of the LTS for Π accessible from $\mathbb{A}_0$.

Analyzing HD. In HD, the relevant starting skeleton is $\mathbb{A}_0$, shown on the left in Fig. 3, where the assumptions—that N was freshly chosen and B's decryption key is uncompromised—are shown in the caption. CPSA identifies the lower

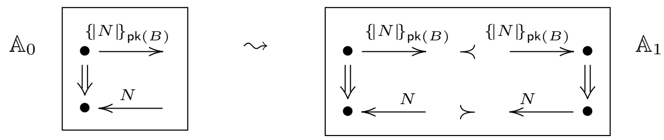

Fig. 3. Goal met by HD, with $\mathsf{unique} = \{N\}, \mathsf{non} = \{\mathsf{pk}(B)^{-1}\}$

reception node as unexplained, given the assumptions: how did N escape from the encryption $\{|N|\}_{\mathsf{pk}(B)}$? It can be solved in only one way, namely, a responder strand can extract N and retransmit it as shown. The result of this step, $\mathbb{A}_1$, is now realized, i.e. fully explained.

Since every realized skeleton that enriches $\mathbb{A}_0$ must solve this test, it must be an enrichment of $\mathbb{A}_1$. In every situation in which an initiator has acted as in $\mathbb{A}_0$, a responder has had a corresponding local session. This is A's authentication guarantee, telling A that B has participated in the session.

Transforming our Analysis under F_1, F_2. Each transformation $F\colon \Pi_1 \mapsto \Pi_2$ determines a map that lifts any skeleton $\mathbb{A}$ of the protocol Π_1 to a corresponding skeleton $F(\mathbb{A})$ of Π_2. In particular, suppose $\mathbb{A}$ contains the first j nodes of a strand s, and s is an instance of a role $\rho_1 \in \Pi_1$. Thus, for some substitution β, $s = \beta(\rho_1)$. When $F(\rho_1) = (\rho_2, g)$, then $F(\mathbb{A})$ should contain the first $g(j)$ nodes of a strand $F(s)$. It should be an instance of $\rho_2 \in \Pi_2$. Specifically $F(s) = \beta'(\rho_2)$, where β' agrees with β on all the parameters appearing in the first j nodes of ρ_1. The remaining parameters of ρ_2 are assigned new values, chosen to be distinct from any of the other values selected for $F(\mathbb{A})$. Since β' depends on all of $\mathbb{A}$, it would be more accurate to write $F_{\mathbb{A}}(s)$ rather than $F(s)$.

If we apply first α_1 and then F_1 mechanically to Fig. 3, we obtain the upper

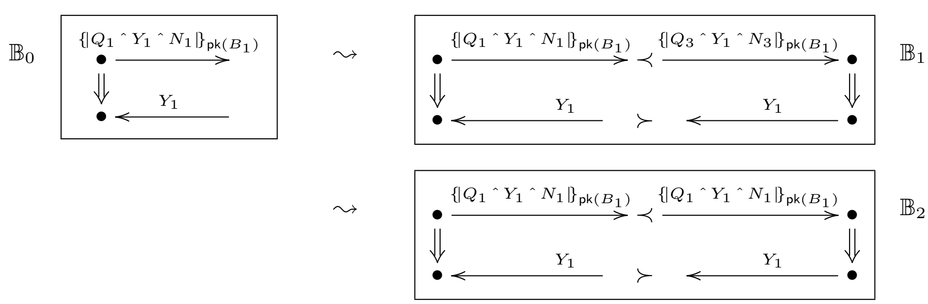

Fig. 4. A goal met by YN, with $\mathsf{unique} = \{Y_1\}, \mathsf{non} = \{\mathsf{pk}(B)^{-1}\}$

two skeletons in Fig. 4. In $\mathbb{B}_0$, the lower node, which is the image of the unsolved test node of $\mathbb{A}_0$, is itself an unsolved test node. Moreover, the new strand in $\mathbb{B}_1$

is the image of the one solution in HD. However, $\mathbb{B}_1$ is not realized. There is only one way to complete the search for a realized skeleton, namely to identify corresponding parameters in the questioner and answerer strands of $\mathbb{B}_1$, setting $Q_3 = Q_1$ and $N_3 = N_1$. The result is $\mathbb{B}_2$.

CPSA generates $\mathbb{B}_2$ from $\mathbb{B}_0$ in one step, which factors through $\mathbb{B}_1$: To realize $\mathbb{B}_0$, one must add the information present in $\mathbb{B}_1$, and also the additional information that $Q_3 = Q_1$ and $N_3 = N_1$. In any YN scenario in which $\mathbb{B}_0$ has occurred, the information in $\mathbb{B}_1$ also holds. Thus, the security analysis of HD has given us sound conclusions information about YN. We now formalize this relation.

4 What is Protocol Analysis?

In our examples, we started with skeletons $\mathbb{A}_0, \mathbb{B}_0$. They were not large enough to be realized, i.e. to be executions that could really happen without any extra regular behavior. We then enriched them to the realized skeletons $\mathbb{A}_1, \mathbb{B}_2$.

Homomorphisms. These enrichments are examples of *homomorphisms* among skeletons [17]. A *homomorphism* $H\colon \mathbb{A} \mapsto \mathbb{B}$ between skeletons of Π transforms messages in a structure-respecting way, mapping transmission nodes to transmission nodes and reception nodes to reception nodes. A homomorphism must preserve the ordering relations of the source, and it must preserve its freshness and non-compromise assumptions. It is an *information-preserving map.*

Homomorphisms determine a preorder, not a partial order, since $\mathbb{A} \xrightarrow{H} \mathbb{B} \xrightarrow{J} \mathbb{A}$ does not imply that $J \circ H$ is the identity. However, if H, J map distinct nodes of their sources injectively to distinct nodes of their targets, then $J \circ H$ is the identity (under reasonable assumptions about the algebra of messages). Thus, these *node-injective* homomorphisms $H\colon \mathbb{A} \mapsto_{ni} \mathbb{B}$ determine a partial order $\leq_{ni}$ on skeletons to within isomorphism.

Lemma 1 ([17, Lemma 3.11]). *$\leq_{ni}$ is a well-founded partial order. Indeed, for every $\mathbb{B}$, there are only finitely many non-isomorphic $\mathbb{A}$ such that $\mathbb{A} \leq_{ni} \mathbb{B}$.*

Protocol analysis: A search through $\mapsto$. 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 skeletons $\mathbb{C}$ such that $\mathbb{A} \mapsto \mathbb{C}$. CPSA computes a set of representative realized skeletons we call *shapes.* 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 $\leq_{ni}$ [12]. 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. Figs. 3 and 4 give examples of test-and-solution steps. The LTS $\rightsquigarrow$ is a subrelation of $\mapsto$.

Indeed, most of the search process works in the partial order $\leq_{ni}$. Although CPSA's implementation is somewhat different, its search could be separated into two phases. After an initial non-node-injective step, all of its test-solving could take place in the node-injective ordering (see [17, Thm. 6.5]).

Core Idea of this Paper. If $F\colon \Pi_1 \mapsto \Pi_2$, the portion of the LTS for Π_1 accessible from $\mathbb{A}_0$ may *simulate* the portion of the LTS for Π_2 accessible from the Π_2 skeleton $F(\mathbb{A}_0)$. Since the labels on the two transition systems may differ, in finding the successful simulation, we freely choose a *relabeling function* G mapping Π_1 labels to Π_2 labels. A simulation using G *progresses* iff, whenever a Π_1 skeleton $\mathbb{A}$ can take some ℓ-transition, the Π_2 skeleton $F(\mathbb{A})$ can take some $G(\ell)$ transition. If for some G, we have a simulation that progresses, then F preserves all security goals concerning the starting situation $\mathbb{A}_0$.

We axiomatize the crucial properties of test-and-solution LTSs, rather than defining one as a function of Π. This has two advantages. First, we can establish goal preservation using finite, often very small, LTSs. Second, particularly for Π_2, we can use a finer LTS (as in Fig. 4) or a coarser one than CPSA would generate.

Definition 2. *Let S be a set of skeletons, and $\mathsf{dead} \in \Lambda$. A ternary relation $\cdot \overset{\cdot}{\rightsquigarrow} \cdot \subseteq S \times \Lambda \times S$ is a* test-and-solution lts *or* TLTS *for S, Λ iff:*

1. *If $\mathbb{A} \in S$, then $\mathbb{A} \rightsquigarrow$ iff $\mathbb{A}$ is not realized;*
2. *If $\mathbb{A} \overset{\ell}{\rightsquigarrow} \mathbb{B}$, then:*
   (a) *If $\ell = \mathsf{dead}$, then $\mathbb{A} = \mathbb{B}$ and there is no realized $\mathbb{C}$ such that $\mathbb{A} \mapsto \mathbb{C}$;*
   (b) *If $\ell \neq \mathsf{dead}$, then $\mathbb{A} \leq_{ni} \mathbb{B}$ and $\mathbb{B} \not\leq_{ni} \mathbb{A}$;*
   (c) *For every homomorphism $J\colon \mathbb{A} \mapsto \mathbb{C}$ from $\mathbb{A}$ to a realized $\mathbb{C}$, there exists some $\mathbb{B}'$ such that $\mathbb{A} \overset{\ell}{\rightsquigarrow} \mathbb{B}'$, and $J = K \circ H$, where $\mathbb{A} \overset{H}{\mapsto} \mathbb{B}' \overset{K}{\mapsto} \mathbb{C}$.*

Let $S(\rightsquigarrow) = \{\mathbb{A} \colon \exists \mathbb{B} \,.\, \mathbb{A} \rightsquigarrow \mathbb{B}\} \cup \{\mathbb{B} \colon \exists \mathbb{A} \,.\, \mathbb{A} \rightsquigarrow \mathbb{B}\}$.

Our TLTSs have the finite image property, i.e. $\{\mathbb{B} \colon \mathbb{A} \overset{\ell}{\rightsquigarrow} \mathbb{B}\}$ is finite for all $\mathbb{A}, \ell$. The non-dead labels in our applications are triples c, E, n_1. In Fig. 3, c is the nonce N. E is the singleton set $\{ \{\!|N|\!\}_{\mathsf{pk}(B)} \}$, which is the one form in which N has been seen. The node n_1 is the lower (reception) node of $\mathbb{A}_0$, in which N is suddenly received outside of its encrypted form in E.

For both steps in Fig. 4, c is the nonce Y_1, and E is the singleton set $\{ \{\!|Q_1 \,\hat{}\, Y_1 \,\hat{}\, N_1|\!\}_{\mathsf{pk}(B_1)} \}$, which is the one form in which N_b has been seen prior to the test nodes n_1. In the first step, the test node n_1 is the lower (reception) node of $\mathbb{B}_0$, in which Y_1 is suddenly received outside of its encrypted form in E. In the second step, the test node n_1 is the upper right (reception) node in $\mathbb{B}_1$, in which Y_1 is received packaged with the (possibly distinct) values Q_3, N_3. The protocol provides no way to perform this repackaging if $Q_3 \neq Q_1$ or $N_3 \neq N_1$, so the only possible explanation is to equate them. Non-singleton Es arise naturally in protocols that use a nonce repeatedly, for successive authentication steps.

Lemma 2. *Suppose that $\cdot \overset{\cdot}{\rightsquigarrow} \cdot$ is a* TLTS, *and $\mathbb{A} \in S(\rightsquigarrow)$. If $\mathbb{A} \mapsto_{ni} \mathbb{C}$ where $\mathbb{C}$ is realized, then there exists a realized $\mathbb{B}$ such that $\mathbb{A} \rightsquigarrow * \,\mathbb{B}$ and $\mathbb{B} \mapsto_{ni} \mathbb{C}$.*

This lemma is an instance of Thm. 1, and can be proved by specializing its proof.

Homomorphisms and Transformations. Homomorphisms are preserved by protocol transformations (Def. 1). Writing $\mathsf{Skel}(\Pi_i)$ for the set of skeletons over Π_i, $F\colon \Pi_1 \to \Pi_2$ determines an *image* operation $\mathsf{Skel}(\Pi_1) \to \mathsf{Skel}(\Pi_2)$. The *image* operation supplies new values for Π_2 role parameters that do not appear in their Π_1 preimages, using some convention. We write $F(\mathbb{A})$ for $\mathbb{A}$'s image.

Lemma 3 ([16]). *Suppose that $F\colon \Pi_1 \to \Pi_2$ is a protocol transformation.*

1. *If $H\colon \mathbb{A} \mapsto \mathbb{B}$, for $\mathbb{A}, \mathbb{B} \in \mathsf{Skel}(\Pi_1)$, there is a unique $F(H)\colon F(\mathbb{A}) \mapsto F(\mathbb{B})$ that commutes with the* image *operation.*
2. *If $G\colon F(\mathbb{A}) \mapsto F(\mathbb{B})$ is any homomorphism between skeletons of this form, then $G = F(H)$ for some $H\colon \mathbb{A} \mapsto \mathbb{B}$.*
3. *If $\mathbb{D} \in \mathsf{Skel}(\Pi_2)$, then $\{\mathbb{A}\colon F(A) \leq_{ni} \mathbb{D}\}$ has a $\leq_{ni}$-maximum in $\mathsf{Skel}(\Pi_1)$.*

5 The Preservation Theorem

Preserving protocol goals is about TLTSs for the two protocols. Assume a relabeling function $G\colon \Lambda(\Pi_1) \times \mathsf{Skel}(\Pi_1) \to \Lambda(\Pi_2)$. The $\mathsf{Skel}(\Pi_1)$ argument determines what parameters in $\mathbb{A}$ to avoid, when choosing new role parameters.

Definition 3. *1. F, G* preserve progress *for $\rightsquigarrow_1$ and $\rightsquigarrow_2$ iff (a) $\ell = \mathsf{dead}$ iff $G(\ell, \mathbb{A}) = \mathsf{dead}$, and (b) for every $\ell \in \Lambda$, $\mathbb{A} \overset{\ell}{\rightsquigarrow}_1$ implies $F(\mathbb{A}) \overset{G(\ell,\mathbb{A})}{\rightsquigarrow}_2$.*

2. *Lts $\rightsquigarrow_1$* simulates *$\rightsquigarrow_2$* under *F, G iff whenever $F(\mathbb{A}) \overset{\ell'}{\rightsquigarrow}_2 \mathbb{B}'$, if $\ell' = G(\ell, \mathbb{A})$, then there exists a $\mathbb{B}$ s.t. $\mathbb{B}' = F(\mathbb{B})$ and $\mathbb{A} \overset{\ell}{\rightsquigarrow}_1 \mathbb{B}$.*

If F, G preserve progress, $\mathbb{A} \in S(\rightsquigarrow_1)$ implies $F(\mathbb{A}) \in S(\rightsquigarrow_2)$. There may be many $\ell' \in \Lambda(\Pi_2)$ outside $\mathrm{ran}(G)$, for instance the second step in Fig. 4. In F_1, F_2, G is determined directly from F_i; in other cases, $G(\ell)$ can use a larger escape set E than the naïve choice suggested by F.

Theorem 1 *Let $F\colon \Pi_1 \to \Pi_2$, and $G\colon \Lambda(\Pi_1) \times \mathsf{Skel}(\Pi_1) \to \Lambda(\Pi_2)$. Let $\rightsquigarrow_1$ and $\rightsquigarrow_2$ be* TLTS*s with $\mathbb{A} \in S(\rightsquigarrow_1) \subseteq \mathsf{Skel}(\Pi_1)$ and $S(\rightsquigarrow_2) \subseteq \mathsf{Skel}(\Pi_2)$. Suppose that:*

1. *F, G preserve progress for $\rightsquigarrow_1$ and $\rightsquigarrow_2$;*
2. *$\rightsquigarrow_1$ simulates $\rightsquigarrow_2$ under F, G.*

For every Π_2-realized $\mathbb{C}$, if $H\colon F(\mathbb{A}) \mapsto_{ni} \mathbb{C}$, there is a Π_1-realized $\mathbb{B}$ such that $\mathbb{A} \rightsquigarrow_1 * \mathbb{B}$, and the accompanying diagram commutes.

$$\begin{tikzcd} F(\mathbb{A}) \arrow[r, "K"'] \arrow[rr, bend left, "H"] \arrow[d, dotted, dash] & F(\mathbb{B}) \arrow[r, "J"'] \arrow[d, dotted, dash] & \mathbb{C} \\ \mathbb{A} \arrow[r, squiggly] & \mathbb{B} & \end{tikzcd}$$

Proof. We use induction on the set $\{\mathbb{D}\colon F(\mathbb{A}) \leq_{ni} \mathbb{D} \leq_{ni} \mathbb{C}\}$, since by Lemma 1, there are only finitely many non-isomorphic $\mathbb{D} \leq_{ni} \mathbb{C}$, and thus only finitely many s.t. $F(\mathbb{A}) \leq_{ni} \mathbb{D} \leq_{ni} \mathbb{C}$.

$\mathbb{A}$ **dead:** If $\mathbb{A} \overset{\mathsf{dead}}{\rightsquigarrow}_1$, then by progress, $F(\mathbb{A}) \overset{\mathsf{dead}}{\rightsquigarrow}_2$ contrary to Def. 2, Clause 2a.
$\mathbb{A}$ **realized:** If $\mathbb{A}$ is realized, it is the desired $\mathbb{B}$, with $K = \mathsf{Id}_{F(\mathbb{A})}$ and $J = H$.

Otherwise, for some $\ell \neq \mathsf{dead}$, $\mathbb{A} \overset{\ell}{\leadsto}_1$. By progress, $F(\mathbb{A}) \overset{\ell'}{\leadsto}_2$ where $\ell' = G(\ell, \mathbb{A})$. By Def. 2, Cl. 2c, H factors through some member of $\{\mathbb{E} \colon F(\mathbb{A}) \overset{\ell'}{\leadsto}_2 \mathbb{E}\}$, say $\mathbb{E}_0$. By simulation, $\mathbb{E}_0 = F(\mathbb{A}')$ for some $\mathbb{A}'$ with $\mathbb{A} \overset{\ell}{\leadsto}_1 \mathbb{A}'$. By Defn. 2, Cl. 2b, $F(\mathbb{A}) \leq_{ni} F(\mathbb{A}')$ but $F(\mathbb{A}') \not\leq_{ni} F(\mathbb{A})$.
Hence, the following proper inclusion eliminates an isomorphism class:

$$\{\mathbb{D} \colon F(\mathbb{A}') \leq_{ni} \mathbb{D} \leq_{ni} \mathbb{C}\} \subsetneq \{\mathbb{D} \colon F(\mathbb{A}) \leq_{ni} \mathbb{D} \leq_{ni} \mathbb{C}\};$$

i.e. the cardinality (modulo isomorphism) is reduced. Thus, we can apply the induction hypothesis to $\mathbb{A}'$ in place of $\mathbb{A}$. □

Lemma 2 is the special case of Thm. 1 in which F, G are the identity functions.

Although Thm. 1 is not about logical formulas, it has a corollary about security goal formulas for Π_1. If the Π_2-realized $\mathbb{C}$ is a *counterexample* to a Π_1 goal formula, then the Π_1-realized $\mathbb{B}$ will also be a counterexample to that goal.

6 The Language $\mathcal{L}(\Pi)$ of a Protocol Π

$\mathcal{L}(\Pi)$ is a classical first order language with equality [15].[2] A formula of the form

$$\forall \overline{x} \,.\, (\phi \quad \supset \quad \exists \overline{y} \,.\, \psi_1 \vee \ldots \vee \psi_j) \tag{1}$$

is a *security goal* if ϕ and each ψ_i is a conjunction of atomic formulas. Null and unary disjunctions in which $j = 0$ or $j = 1$ are permitted. We assume that $\overline{x}$ and $\overline{y}$ are disjoint lists of variables, and that all variables free in any ψ_i but not in ϕ appear in $\overline{y}$. Authentication and confidentiality goals do take the form (1).

$\mathcal{L}(\Pi)$ says nothing about the structure of Π's messages, so it can represent goals that are preserved when that message structure is transformed. It describes nodes by their role, their index along the role, and the role's parameters. $\mathcal{L}(\Pi)$ contains function symbols $\mathtt{pk}(a)$, $\mathtt{sk}(a)$, and $\mathtt{inv}(a)$ denoting a's public encryption key; a's private signature key; and the inverse member of a key pair, resp.

The predicates in $\mathcal{L}(\Pi)$ (other than equality) are grouped into five kinds. The first two kinds are identical for all protocols. We give the predicates an entirely classical semantics by specifying when each predicate is satisfied in a model (skeleton) $\mathbb{A}$, relative to a variable assignment η.

Order. There is one order predicate, named $\mathtt{Prec}(n, m)$, and $\mathbb{A} \models_\eta \mathtt{Prec}(n, m)$ iff $\eta(n)$ and $\eta(m)$ are nodes in $\mathbb{A}$, and $\eta(n) \prec_{\mathbb{A}} \eta(m)$.

Security Assumptions. The three security assumption predicates are $\mathtt{Non}(v)$, $\mathtt{Unq}(v)$, and $\mathtt{UnqAt}(n, v)$. $\mathbb{A} \models_\eta \mathtt{Non}(v)$ iff $\eta(v) \in \mathsf{non}_{\mathbb{A}}$. $\mathbb{A} \models_\eta \mathtt{Unq}(v)$ iff $\eta(v) \in \mathsf{unique}_{\mathbb{A}}$. $\mathbb{A} \models_\eta \mathtt{UnqAt}(n, v)$ iff (i) $\eta(v) \in \mathsf{unique}_{\mathbb{A}}$; (ii) $\eta(n)$ is a node in $\mathbb{A}$; and (iii) $\eta(v)$ originates at $\eta(n)$.

Position. If the longest of the (finitely many) roles in the protocol Π is of length k, then there are k position predicates, $\mathtt{AtPos1}(s, n), \ldots, \mathtt{AtPos}k(s, n)$. $\mathbb{A} \models_\eta \mathtt{AtPos}i(s, n)$ iff (i) $\eta(s)$ is a strand; (ii) $\eta(n) \in \mathbb{A}$; and (iii) $n = s \downarrow i$.

[2] Although the syntax is simplified from [15], $\mathcal{L}(\Pi)$'s expressiveness is unchanged.

Role. If Π contains j roles, then there are j role predicates. If $\texttt{RoleName}(s)$ is the role predicate associated with role $\rho \in \Pi$, then $\mathbb{A} \models_\eta \texttt{RoleName}(s)$ iff (i) $\eta(s)$ is a strand; (ii) at least one node of $\eta(s)$ is in $\mathbb{A}$; and (iii) the portion of s within $\mathbb{A}$ agrees with an initial segment of an instance of ρ.

Parameter. There is a set of predicates $\texttt{ParamName}(s, v)$. For each role $\rho \in \Pi$, we have an injective association from the parameters of ρ to some of these predicates. Different roles may share predicates. $\mathbb{A} \models_\eta \texttt{ParamName}(s, t)$ holds if (i) $\eta(s)$ is a strand; (ii) at least one node of $\eta(s)$ is in $\mathbb{A}$; and (iii) there exist ρ, α, v s.t. (a) s and $s' = \alpha(\rho)$ agree on the portion within $\mathbb{A}$; (b) ρ's parameter v is associated with $\texttt{ParamName}$; (c) the portion of ρ within $\mathbb{A}$ is long enough to have an occurrence of v; and (d) $\alpha(v) = \eta(t)$. Moreover, any other ρ', α', a' satisfying (iii a–c) should also satisfy (iii d).

The definition ensures that $\mathbb{A} \models_\eta \phi[s]$ is never sensitive to the part of $\eta(s)$ outside $\mathbb{A}$. As in Def. 1, Clause 5, $\mathcal{L}(\Pi)$ is insensitive to not-yet-executed branch points.

Example 1: $\mathcal{L}$(HD). $\mathcal{L}(\mathsf{HD})$ contains the shared vocabulary, and the predicates:

$$\begin{array}{cccc} \texttt{AtPos1}(s,n) & \texttt{AtPos2}(s,n) & \texttt{Init}(s) & \texttt{Resp}(s) \\ \texttt{Peer}(s,b) & \texttt{Nonce}(s,v) & \texttt{Self}(s,b) & \end{array}$$

We use $\texttt{Nonce}(s, v)$ for both initiator and responder strands, but it seems safer to distinguish the initiator's intended peer B—if, hopefully, the corresponding private decryption key is uncompromised—from the responder's actual identity. For the skeleton $\mathbb{A}_0$ of Fig. 3, and the formula $\Phi_0 =$

$$\begin{array}{l} \texttt{Init}(s) \wedge \texttt{AtPos1}(s,n) \wedge \ \texttt{AtPos2}(s,m) \ \wedge \ \texttt{Peer}(s,b) \\ \qquad \wedge \ \texttt{Nonce}(s,v) \ \wedge \texttt{Non}(\texttt{inv}(\texttt{pk}(b))) \wedge \texttt{UnqAt}(n,v), \end{array}$$

we have $\mathbb{A}_0 \models_\eta \Phi_0$ where η is the assignment sending the variable s to the initiator strand shown; sending n and m to its first and second nodes resp.; sending b to the name B; and sending v to nonce N. Letting $\Phi_1 =$

$$\begin{array}{l} \texttt{Resp}(s') \wedge \texttt{AtPos1}(s',n') \wedge \texttt{AtPos2}(s',m') \wedge \ \texttt{Self}(s',b) \\ \qquad \wedge \ \texttt{Nonce}(s',v) \ \wedge \ \texttt{Prec}(n,n') \ \wedge \texttt{Prec}(m',m), \end{array}$$

we have $\mathbb{A}_1 \models_\theta \Phi_0 \wedge \Phi_1$, where θ agrees with η for the variables mentioned, and moreover θ sends s' to the responder strand shown; and sends n', m' to its first and second nodes resp. The variables b, v are *not* primed in Φ_1, expressing the agreement of the initiator and responder strands on these parameters.

Φ_0 is satisfied in both $\mathbb{A}_0$ and $\mathbb{A}_1$. Indeed, because Φ_0 is a conjunction of atoms, it will be satisfied in *every* homomorphic image of $\mathbb{A}_0$. Specifically, if $H : \mathbb{A}_0 \to \mathbb{C}$, then composing H with the variable assignment η, we have $\mathbb{C} \models_{H \circ \eta} \Phi_0$. Moreover, this is exact: If $\mathbb{C} \models_\theta \Phi_0$, then for some $H : \mathbb{A}_0 \to \mathbb{C}$, $\theta = H \circ \eta$.

Definition 4. $\mathbb{A}, \eta$ *is a* characteristic pair *for Φ iff $\mathbb{A} \models_\eta \Phi$ and, for all $\mathbb{B}, \theta$,*

$$\mathbb{B} \models_\theta \Phi \quad \supset \quad \exists! H \, . \, H : \mathbb{A} \to \mathbb{B} \ \textit{and} \ \theta \restriction \mathsf{fv}(\Phi) = H \circ \eta. \tag{2}$$

$\mathbb{A}$ *is a* characteristic skeleton *for Φ iff, $\exists \eta \, . \, \mathbb{A}, \eta$ is a characteristic pair for Φ.*

We write $\mathbb{A}, \eta = \mathsf{cp}(\Phi)$ for the characteristic pair and $\mathbb{A} = \mathsf{cs}(\Phi)$ for the characteristic skeleton, when they exist, since they are unique to within isomorphism. $\mathbb{A}_0$ is the characteristic skeleton of Φ_0, i.e. $\mathbb{A}_0 = \mathsf{cs}(\Phi_0)$, and $\mathbb{A}_1 = \mathsf{cs}(\Phi_0 \wedge \Phi_1)$. In Fig. 3, since every $H\colon \mathbb{A}_0 \mapsto \mathbb{C}$ to a realized skeleton factors through $\mathbb{A}_0 \rightsquigarrow \mathbb{A}_1$, we have demonstrated the goal $\Gamma_1 =$

$$\forall s, n, m, b, v\,.\ (\Phi_0 \quad \supset \quad \exists s', n', m'\,.\ \Phi_1). \tag{3}$$

Implicit Typing. A variable s appearing in a role predicate in ϕ, or as the first argument to a position predicate or a role parameter predicate, is called a *strand variable* in ϕ. A variable n appearing as the second argument to a role position predicate, or either argument to an order predicate, or the first argument to an `UnqAt` predicate, is called a *node variable*.

A conjunction of atoms ϕ is *implicitly typed* if strand and node variables are disjoint; every strand variable appears in exactly one role predicate; and every node variable appears in exactly one position predicate. Φ_0 and $\Phi_0 \wedge \Phi_1$ are implicitly typed, but Φ_1 alone is not. Node variables m, n occur in no position predicate in Φ_1. By associativity and commutativity, an implicitly typed ϕ can be rewritten so every subformula is implicitly typed: the role predicate for s can precede position and parameter predicates for it, and the position predicate for n can precede `Pred` and `UnqAt` predicates for it. As in [15, Thm. 5.2]:

Lemma 4. *If ϕ is implicitly typed, the characteristic skeleton $\mathsf{cs}(\phi)$ is defined.*

If Γ is a goal formula as in Eqn. (1), then we say that Γ is implicitly typed if ϕ is, and each of the conjunctions $\phi \wedge \psi_i$ is, where $1 \leq i \leq j$.

Example 2: $\mathcal{L}$(YN). The language $\mathcal{L}(\mathsf{YN})$, like $\mathcal{L}(\mathsf{HD})$, has the position predicates $\texttt{AtPos1}(s, n)$ and $\texttt{AtPos2}(s, n)$. It has four role predicates, namely $\texttt{QAf}(s)$, $\texttt{QNg}(s)$, $\texttt{AnAf}(s)$, and $\texttt{AnNg}(s)$. Again expressing an intended peer via $\texttt{Peer}(s, b)$, and an actual identity via $\texttt{Self}(s, b)$, we have five parameter predicates:

$$\texttt{Quest}(s, q), \quad \texttt{YesVal}(s, v), \quad \texttt{NoVal}(s, v), \quad \texttt{Peer}(s, b), \quad \texttt{Self}(s, b).$$

Protocol Transformations and Language Translations. Each $F\colon \Pi_1 \to \Pi_2$ determines a translation $Tr_F(\cdot)$ between implicitly typed goal formulas of $\mathcal{L}(\Pi_1)$ and $\mathcal{L}(\Pi_2)$. We translate conjunctions one atomic formula at a time. Let $F(\rho_1) = (\rho_2, g)$. The order and assumption predicates are translated verbatim.

$\texttt{RolePred1}(s)$**:** If `RolePred1` is the role predicate for ρ_1 and `RolePred2` is the role predicate for $\rho_2 \in \Pi_2$, the result is $\texttt{RolePred2}(s)$.

$\texttt{PosPred}i(s, n)$**:** By the assumed typing, there is a single conjunct $\texttt{RolePred1}(s)$ with the same s. If `RolePred1` is the role predicate for ρ_1, then letting the index $j = g(i)$, the result is $\texttt{PosPred}j(s, n)$.

$\texttt{ParamName1}(s, t)$**:** Again, there is a single conjunct $\texttt{RolePred1}(s)$ with the same s, by the assumed typing. Suppose `RolePred1` is the role predicate for ρ_1, and ρ_1 associates parameter a with `ParamName1`. Select the predicate `ParamName2` that ρ_2 associates with a. The result is $\texttt{ParamName2}(s, t)$.
If either `ParamName1` is not associated with any parameter of ρ_1, or a does not appear in ρ_2, then the result is vacuously true, e.g. $s = s \wedge t = t$.

The choice of parameters in Π_1 and Π_2, together with the per-role associations of parameters to predicates, compose to form a map from parameter predicates of $\mathcal{L}(\Pi_1)$ to parameter predicates of $\mathcal{L}(\Pi_2)$. A renaming such as α_1, which forms $\alpha_1(\mathsf{HD})$ as the source protocol for F_1, readjusts this composed mapping from parameter predicates of $\mathcal{L}(\Pi_1)$ to parameter predicates of $\mathcal{L}(\Pi_2)$.

If $\phi \wedge \psi_i$ is implicitly typed, $Tr^{\phi}_{F}(\psi)$ translates ψ_i the same way, using conjuncts of ϕ to provide the implicit typing. We have ensured that $\mathsf{fv}(Tr_F(\phi)) = \mathsf{fv}(\phi)$ and $\mathsf{fv}(Tr^{\phi}_{F}(\psi)) = \mathsf{fv}(\psi)$. So, let $Tr_F(\forall \overline{x} \,.\, (\phi \supset \exists \overline{y} \,.\, \psi_1 \vee \ldots \vee \psi_k))$ be

$$\forall \overline{x} \,.\, (Tr_F(\phi) \supset \exists \overline{y} \,.\, Tr^{\phi}_{F}(\psi_1) \vee \ldots \vee Tr^{\phi}_{F}(\psi_k)). \tag{4}$$

For the goal formula Γ_1 describing Fig. 3, $Tr_{F_1}(\Gamma_1)$ is:

$$\begin{array}{lllll} \forall s,n,m,b,v \,.\; (\mathtt{QAf}(s) \wedge & \mathtt{AtPos1}(s,n) & \wedge\, \mathtt{AtPos2}(s,m) \wedge & \mathtt{Peer}(s,b) & \\ \qquad\qquad\qquad\qquad \wedge\, \mathtt{Non}(\mathtt{inv}(\mathtt{pk}(b))) \wedge & \mathtt{YesVal}(s,v) & \wedge\; \mathtt{UnqAt}(n,v) & & \\ \supset \exists s',n',m' \,.\; \mathtt{AnAf}(s') \wedge & \mathtt{AtPos1}(s',n') & \wedge\, \mathtt{AtPos2}(s',m') \wedge & \mathtt{Self}(s',b) & \\ \qquad\qquad\qquad\qquad \wedge & \mathtt{YesVal}(s',v) & \wedge\; \mathtt{Prec}(n,n') & \wedge\, \mathtt{Prec}(m',m)). & \end{array}$$

If η is an variable assignment taking values in $\mathbb{A}$, let $\overline{\eta}$ be the corresponding assignment taking values in $F(\mathbb{A})$, so that $\overline{\eta}$ is the composition of η with the "image" map from $\mathbb{A}$ to $F(\mathbb{A})$.

Lemma 5. *1. If $\mathbb{A} \models_{\eta} \phi$, then $F(\mathbb{A}) \models_{\overline{\eta}} Tr_F(\phi)$.*
2. If $\mathbb{A} \models_{\eta} \phi \wedge \psi$, then $F(\mathbb{A}) \models_{\overline{\eta}} Tr^{\phi}_{F}(\psi)$.
3. If $\mathbb{B} \models_{\theta} Tr_F(\phi)$ and $\mathsf{cp}(\phi) = \mathbb{A},\eta$, then there exists a J such that $J\colon F(\mathsf{cs}(\phi)) \to \mathbb{B}$, and θ agrees with $J \circ \overline{\eta}$ on $\mathsf{fv}(\phi)$.
4. When $\mathsf{cs}(\phi)$ exists, so does $\mathsf{cs}(Tr_F(\phi))$, and $F(\mathsf{cs}(\phi)) = \mathsf{cs}(Tr_F(\phi))$.

Proof. See Appendix A.2.

7 Preserving Security Goal Formulas

Π achieves a goal formula Γ iff, for all Π-realized skeletons $\mathbb{C}$, $\mathbb{C} \models \Gamma$.

Theorem 2 *Suppose that Π_1 achieves a goal $\Gamma = \forall \overline{x} \,.\, (\phi \supset \exists \overline{y} \,.\, \psi_1 \vee \ldots \psi_j)$; that $\mathsf{cp}(\phi) = \mathbb{A},\eta$; and that $F\colon \Pi_1 \to \Pi_2$. Then Π_2 achieves $Tr_F(\Gamma)$, assuming there exists a G and* TLTSs *$\rightsquigarrow_1$ and $\rightsquigarrow_2$ as in Thm. 1, i.e. $\mathbb{A} \in S(\rightsquigarrow_1)$ and*

1. *F, G preserve progress for $\rightsquigarrow_1$ and $\rightsquigarrow_2$;*
2. *$\rightsquigarrow_1$ simulates $\rightsquigarrow_2$ under F, G.*

Proof sketch. Suppose that $\mathbb{C}$ is a Π_2-realized skeleton such that $\mathbb{C} \models_{\theta} Tr_F(\phi)$. Since (Lemma 5, Clause 4) $\mathsf{cs}(Tr_F(\phi)) = F(\mathbb{A})$, we have $H\colon F(\mathbb{A}) \to \mathbb{C}$.

If H is node-injective, we apply Thm. 1 to infer that $F(\mathbb{A}) \xrightarrow{K}_{ni} F(\mathbb{B}) \xrightarrow{J}_{ni} \mathbb{C}$, where $\mathbb{A} \rightsquigarrow_1 * \mathbb{B}$, and $\mathbb{B}$ is Π_1-realized. Since Π_1 achieves Γ, $\mathbb{B} \models_{\zeta} \psi_i$ for some i^{th} disjunct. By Lemma 5, Clause 2, $F(\mathbb{B}) \models_{\overline{\zeta}} Tr^{\phi}_{F}(\psi_i)$. Since J preserves conjunctions, $\mathbb{C} \models_{J \circ \overline{\zeta}} Tr^{\phi}_{F}(\psi_i)$. Hence $\mathbb{C} \models_{\theta} \exists \overline{y} \,.\, Tr^{\phi}_{F}(\psi_i)$.

In App. A.2, we ensure that θ and $J \circ \overline{\zeta}$ agree on the variables $\overline{x}$.

If H is not node-injective, then we split $H = H_0 \circ K_0$, where K_0 is node-surjective and H_0 is node-injective, and apply this argument to H_0. $\square$

Related Work The safe protocol transformation problem is not new. In a key special case, "protocol composition," with $\Pi_1 \subseteq \Pi_2$, it dates from the 1990s [21]. A strong form of composition is reactive simulatability [25, 3] or universal composability [6]; weaker forms may still be cryptographically justified [11].

In the symbolic model, we provided a widely applicable and practically useful criterion [18, 13]. Cortier et al.'s criterion is in some ways broader but in other ways narrower [7]; cf. [1]. Our [14] covers the union of [18, 7, 1]. We here generalize [14] beyond the composition case where $\Pi_1 \subseteq \Pi_2$.

The Protocol Composition Logic PCL considers refinements that preserve security goals [10, 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. [10]'s "parallel" and "sequential" composition amounts to $\Pi_1 \subseteq \Pi_2$. Their "protocol refinement using templates" [9] suggested many of our examples. By contrast with Distributed Temporal Logic [5], $\mathcal{L}(\Pi)$ is intended to express less about messages. However, satisfaction is undecidable. Lowe and Auty [23] refine protocols to concrete messages starting from formulas in a Hoare-like logic that represent the effect of messages. Maffei et al. [2] express the effects of messages by abstract tags, and provide constraints on instantiating the tags by concrete messages.

"Protocol compilers" transform their input automatically. Some start with a crypto-free protocol, and transform it into a protocol meeting security goals [8, 4]. Others transform a protocol secure in a weak adversary model into protocols satisfying those goals with multi-session, active adversary [20].

Future work. We leave a major gap: What syntactic property of $F\colon \Pi_1 \to \Pi_2$ ensures that F preserves security goals? A clue comes from the "disjoint encryption" property [18, 14], cf. [23, 7]. 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 TLTS 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. I am grateful to Dan Dougherty, John Ramsdell, Paul Rowe, and Javier Thayer. The simplification of $\mathcal{L}(\Pi)$ vs. [15] arose from a conversation with Ramsdell. Thanks to Siraj Sayani and Soumentra Ghosal, whose hospitality I enjoyed in Coonoor while writing a good part of this paper.

References

1. 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.

2. 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.
3. Michael Backes, Birgit Pfitzmann, and Michael Waidner. A universally composable cryptographic library. Available at `http://eprint.iacr.org/2003/015/`, 2003.
4. 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.
5. 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.
6. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Report 2000/067, International Association for Cryptographic Research, October 2001. Extended Abstract appeared in proceedings of the 42nd Symposium on Foundations of Computer Science (FOCS), 2001.
7. Véronique Cortier, Jérémie Delaitre, and Stéphanie Delaune. Safely composing security protocols. In V. Arvind and Sanjiva Prasad, editors, *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, LNCS, New Delhi, India, December 2007. Springer.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. Joshua D. Guttman. Authentication tests and disjoint encryption: a design method for security protocols. *Journal of Computer Security*, 12(3/4):409–433, 2004.
14. 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, 2009.
15. Joshua D. Guttman. Security theorems via model theory. *EXPRESS: Expressiveness in Concurrency (EPTCS)*, 8:51, 2009. doi:10.4204/EPTCS.8.5.
16. 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.
17. 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.
18. Joshua D. Guttman and F. Javier Thayer. Protocol independence through disjoint encryption. In *Computer Security Foundations Workshop*. IEEE CS Press, 2000.

19. 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.
20. Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. *J. Cryptology*, 20(1):85–113, 2007.
21. John Kelsey, Bruce Schneier, and David Wagner. Protocol interactions and the chosen protocol attack. In *Security Protocols Workshop*. Springer, 1998.
22. Leslie Lamport. Time, clocks and the ordering of events in a distributed system. *CACM*, 21(7):558–565, 1978.
23. Gavin Lowe and Michael Auty. A calculus for security protocol development. Technical report, Oxford University Computing Laboratory, March 2007.
24. Roger Needham and Michael Schroeder. Using encryption for authentication in large networks of computers. *CACM*, 21(12), December 1978.
25. 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.
26. John D. Ramsdell and Joshua D. Guttman. CPSA: A cryptographic protocol shapes analyzer. In *Hackage*. The MITRE Corporation, 2009. `http://hackage.haskell.org/package/cpsa`; see esp. `doc` subdirectory.

A Appendices

In this appendix, we first introduce a few additional transformations, to indicate that they are a flexible and natural set of relations among protocols. We then give a proof of Lemma 5 and some more detail on the proof of Thm. 2. [17], although not yet printed, is available at URL

`http://web.cs.wpi.edu/~guttman/pubs/ssg.pdf`.

A.1 Some Additional Transformations

An Enriched Yes-or-No Protocol. We can illustrate three additional protocol transformations F_3, F_4, F_5 by considering an enriched version YN^+ of YN. YN^+ is intended to preserve the goals of YN, and to achieve an additional goal, namely that the Answerer authenticates his Questioner via an HD-like mechanism. The Answerer predistributes a token T to the Questioner. The latter includes T with its query. This allows the Answerer to bill per question for its services, or to reject questions from non-subscribers. We show the affirmative case in Fig. 5; the negative case is symmetrical, using the other nonce in the last message. This protocol may be regarded as the result of three transformations.

F_3 sends $\mathrm{QAf}_{\mathsf{YN}}$ to $\mathrm{QAf}_{\mathsf{YN}^+}$, $\mathrm{QNg}_{\mathsf{YN}}$ to $\mathrm{QNg}_{\mathsf{YN}^+}$, $\mathrm{AnAf}_{\mathsf{YN}}$ to $\mathrm{AnAf}_{\mathsf{YN}^+}$, and $\mathrm{AnNg}_{\mathsf{YN}}$ to $\mathrm{AnNg}_{\mathsf{YN}^+}$. However, F_3 sends the first and second nodes of each source role to the *second* and *third* nodes of the corresponding target role. F_3 is responsible for ensuring that YN^+ preserves the goals of YN.

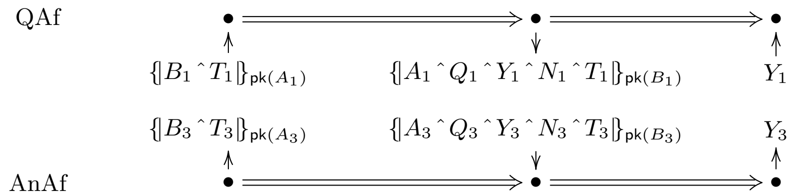

Fig. 5. The Enriched Yes-or-No Protocol YN^+ (Affirmative Half)

F_4 is a transformation from $\alpha_4(\mathsf{HD})$ to YN^+, where

$$\alpha_4 = [N \mapsto T_3,\ N' \mapsto T_1,\ B \mapsto A_3,\ B' \mapsto A_1].$$

It flips roles to provide authentication in the opposite direction. F_4 maps $\mathrm{Init}_{\alpha_4(\mathsf{HD})}$ to $\mathrm{AnAf}_{\mathsf{YN}^+}$, sending the first and second nodes to the first and second nodes respectively. F_4 maps $\mathrm{Resp}_{\alpha_4(\mathsf{HD})}$ to $\mathrm{QAf}_{\mathsf{YN}^+}$, again with the same mapping of nodes. F_4 ensures that B authenticates the questioner A whenever giving an affirmative answer.

F_5 is a transformation, symmetric with F_4, with the corresponding negative roles as targets. F_5 ensures that B authenticates the questioner A whenever giving a negative answer.

F_3 illustrates node index functions that are not the identity. Corresponding to the analysis $\mathbb{B}_0 \rightsquigarrow \mathbb{B}_2$ (see Fig. 4), which we will now regard as a single step, we have the first step shown in Fig. 6, where $\mathbb{C}_0 = F(\mathbb{B}_0)$ and $\mathbb{C}_2 = F(\mathbb{B}_2)$, and $\mathbb{C}_3$ is the shape for $\mathbb{C}_0$.

F_4, when applied to the analysis in Fig. 3, yields the first step shown in Fig. 7. Here, $\mathbb{D}_0 = F_4(\alpha_4(\mathbb{A}_0))$ and $\mathbb{D}_1 = F_4(\alpha_4(\mathbb{A}_1))$. On the assumption given here, that $\mathsf{non} = \{\mathsf{pk}(A_3)^{-1}\}$, we can take a further step to identify the message sent by B_3 with the message received by $A_1 = A_3$, therefore ensuring that A_3 agrees on the identity B_3. The other parameters need not agree unless $\mathsf{pk}(B_3)^{-1} \in \mathsf{non}$ also. The analysis of F_5 is identical (to within renaming), as the roles QAf and QNg have not diverged up to their second node, nor have AnAf and AnNg.

The Protocol NSL^-. NSL^- is a slight variant of Needham-Schroeder-Lowe, which, as we will show later, is a goal-preserving image of HD in two different ways. NSL^- authenticates each of the two participants, using an HD-like mechanism. We have arranged the two roles in Fig. 8 to emphasize their symmetry.

Transformations from HD to NSL^-. We can regard NSL^- as an image of HD by mapping the the two nodes of the HD initiator to the first two nodes of the NSL^- initiator. Likewise, we carry the two nodes of the HD responder to the first two nodes of the NSL^- responder. It uses the renaming $\alpha_6 = [N \mapsto N_a, N' \mapsto N'_a]$. We call this transformation F_6.

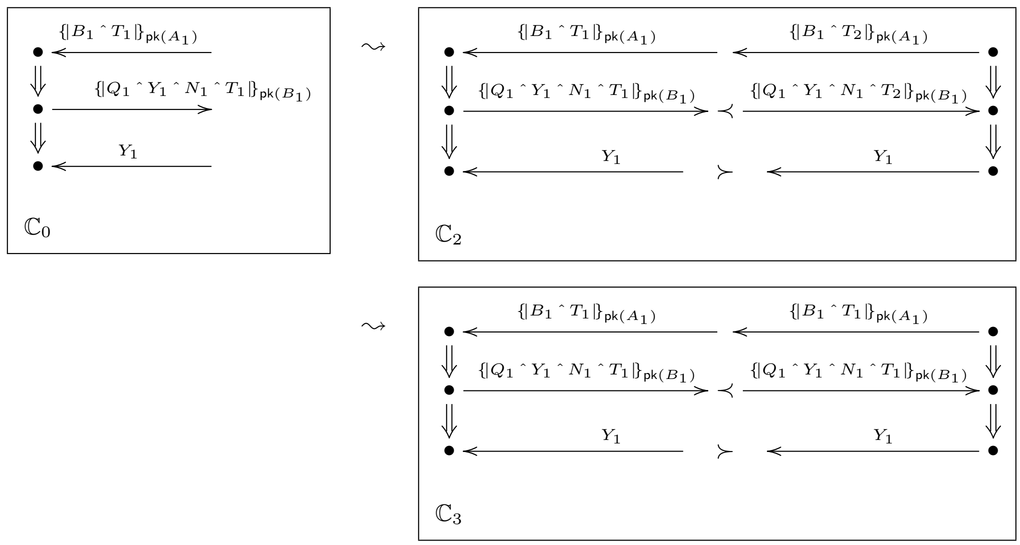

Fig. 6. Analysis for YN^+ Questioner, with $\mathsf{unique} = \{Y_1\}, \mathsf{non} = \{\mathsf{pk}(B)^{-1}\}$

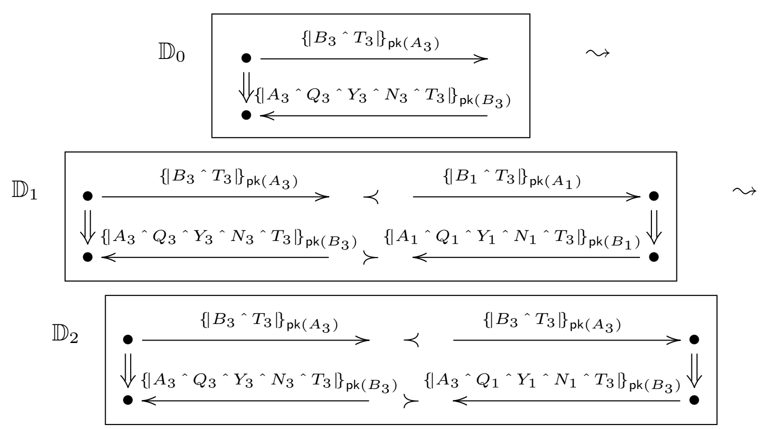

Fig. 7. Analysis for YN^+ Answerer, $\mathsf{unique} = \{T_3\}, \mathsf{non} = \{\mathsf{pk}(A_3)^{-1}\}$

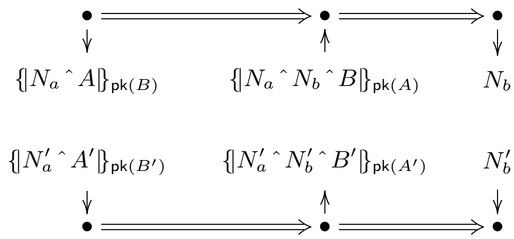

Fig. 8. NSL^-: Needham-Schroeder-Lowe minus encryption in last message

For our second transformation, we flip the roles, and send the initiator role of HD to the responder role of NSL^-, and vice versa. We use a renaming $\alpha_7 = [N \mapsto N_b, N' \mapsto N'_b, B \mapsto A, B' \mapsto A']$.

We now define a mapping from $\alpha_7(\mathsf{HD})$ to NSL^-. We map the two nodes of the $\alpha_7(\mathsf{HD})$ initiator to the *second* and *third* node of the NSL^- responder role. We send the two nodes of the $\alpha_7(\mathsf{HD})$ responder to the second and third node of the NSL^- initiator role. We call this transformation F_7.

The preservation of the analysis holds for F_6, F_7 as before.

NSL^- is itself a suitable starting point for further transformations, including an electronic commerce EPMO we have designed [19]. Indeed, a core motivation for the present line of work was to give a systematic explanation of the design heuristics that we have used in constructing EPMO and other protocols.

These examples are intended to illustrate the fact that many progressively more complex protocols can be built systematically and safely using these methods. See also [9].

A.2 Proofs of Lemma 5 and Theorem 2

Given $F\colon \Pi_1 \to \Pi_2$ and $\mathbb{A} \in \mathsf{Skel}(\Pi_1)$, define $\overline{\eta}$ to be the assignment θ s.t.:

1. If $\eta(v)$ is not a strand or node in $\mathbb{A}$, then $\overline{\eta}(v) = \eta(v)$; and
2. If $\eta(v)$ is a strand or node in $\mathbb{A}$, then $\overline{\eta}(v)$ is the image of $\eta(v)$ in $F(\mathbb{A})$.

Lemma 5. *Let $F\colon \Pi_1 \to \Pi_2$; let $\phi, \phi \wedge \psi$ be implicitly typed conjunctions.*

1. *If $\mathbb{A} \models_\eta \phi$, then $F(\mathbb{A}) \models_{\overline{\eta}} Tr_F(\phi)$.*
2. *If $\mathbb{A} \models_\eta \phi \wedge \psi$, then $F(\mathbb{A}) \models_{\overline{\eta}} Tr_F^{\phi}(\psi)$.*
3. *If $\mathbb{B} \models_\theta Tr_F(\phi)$ and $\mathsf{cp}(\phi) = \mathbb{A}, \eta$, then there exists a J s.t. $J\colon F(\mathsf{cs}(\phi)) \to \mathbb{B}$, and θ agrees with $J \circ \overline{\eta}$ on $\mathsf{fv}(\phi)$.*
4. *When $\mathsf{cs}(\phi)$ exists, so does $\mathsf{cs}(Tr_F(\phi))$, and $F(\mathsf{cs}(\phi)) = \mathsf{cs}(Tr_F(\phi))$.*

Proof. **1.** We assume that ϕ is written in the order defined immediately before Lemma 4, so that we may assume that every subformula of ϕ is also implicitly typed. We argue by structural induction on ϕ.
Base case: If ϕ is the trivially true conjunction with 0 conjuncts, then $Tr_F(\phi)$ is also the null conjunction, which is true in every structure.

Induction step: If ϕ is $\phi_1 \wedge \phi_2$, where ϕ_2 is an atomic formula, suppose that the claim already holds of ϕ_1.

Let ϕ_2 be a role predicate $\texttt{RoleName}(s)$. Then $\eta(s)$ is a strand with at least one node in $\mathbb{A}$, and all of its nodes in $\mathbb{A}$ agree with an initial segment of an instance of the associated role ρ_1. Thus, its F-image in $F(\mathbb{A})$ has at least $g(1)$ nodes in $\mathbb{A}$ and all of its nodes in $\mathbb{A}$ agree with an initial segment of an instance of the corresponding role ρ_2. Hence $F(\mathbb{A}) \models_{\overline{\eta}} \mathit{Tr}_F(\phi_2)$. Position predicates are similar.

Let ϕ_2 be a parameter predicate $\texttt{ParamName}(s,t)$. Since ϕ is implicitly typed, there is just one $\texttt{RoleName}(s)$ with the same s in ϕ_1, and $\mathbb{A} \models_\eta \texttt{RoleName}(s)$. Thus, $\mathit{Tr}_F(\phi_2)$ is the F-corresponding parameter name predicate, and the F-image of $\eta(s)$ has the same parameter $\eta(t)$. Hence, $F(\mathbb{A}) \models_{\overline{\eta}} \mathit{Tr}_F(\phi_2)$.

Order and assumption predicates and $=$ are immediate from the definitions.

2. $\mathit{Tr}_F(\phi \wedge \psi)$ entails $\mathit{Tr}^{\phi}_F(\psi)$, and Clause 1 implies $F(\mathbb{A}) \models_{\overline{\eta}} \mathit{Tr}_F(\phi \wedge \psi)$.

3. By Lemma 4, $\mathsf{cs}(\phi)$ exists. By the properties of cs [15], each strand with nodes in $\mathbb{A}$ is $\eta(s)$ for some distinct s in a role predicate in ϕ. If $\mathbb{A}$ contains i nodes of $\eta(s)$, then there is a node position predicate for i, s, and some n. Moreover, each parameter to the associated role ρ takes an atom or indeterminate as its value in $\eta(s)$, not a concatenation or encryption.

We will construct a $J \colon F(\mathsf{cs}(\phi)) \to \mathbb{B}$. $J = [\phi, \alpha]$ is defined: $\phi(\overline{\eta}(s)) = \theta(s)$. If a is a parameter to $\overline{\eta}(s)$, then $\alpha(a)$ is the corresponding parameter to $\overline{\theta}(s)$. By the universality of $\mathsf{cs}(\phi)$ and of $F(\mathbb{A})$, $J = [\phi, \alpha]$ is a homomorphism.

4. By the syntax, $\mathit{Tr}_F(\phi)$ is implicitly typed, so Lemma 4 implies $\mathsf{cs}(\mathit{Tr}_F(\phi))$ exists. By the previous clause, $J \colon F(\mathsf{cs}(\phi)) \to \mathsf{cs}(\mathit{Tr}_F(\phi))$.

By clause 1, $F(\mathsf{cs}(\phi)) \models_{\overline{\eta}} \mathit{Tr}_F(\phi)$. Thus, Def. 4 entails that $K \colon \mathsf{cs}(\mathit{Tr}_F(\phi)) \to F(\mathsf{cs}(\phi))$. Hence, by the uniqueness in Def. 4, $J \circ K = \mathsf{Id}$. Hence, $F(\mathsf{cs}(\phi))$ and $\mathsf{cs}(\mathit{Tr}_F(\phi))$ are isomorphic. $\square$

Theorem 2. *Suppose that Π_1 achieves a goal $\Gamma = \forall \overline{x} \,.\, (\phi \supset \exists \overline{y} \,.\, \psi_1 \vee \ldots \psi_j)$; that $\mathsf{cp}(\phi) = \mathbb{A}, \eta$; and that $F \colon \Pi_1 \to \Pi_2$. Then Π_2 achieves $\mathit{Tr}_F(\Gamma)$, assuming there exists a G and* TLTSs $\rightsquigarrow_1$ *and* $\rightsquigarrow_2$ *as in Thm. 1, i.e. $\mathbb{A} \in S(\rightsquigarrow_1)$ and*

1. *F, G preserve progress for $\rightsquigarrow_1$ and $\rightsquigarrow_2$;*
2. *$\rightsquigarrow_1$ simulates $\rightsquigarrow_2$ under F, G.*

Proof. $\mathit{Tr}_F(\Gamma)$ is $\forall \overline{x} \,.\, (\mathit{Tr}_F(\phi) \supset \exists \overline{y} \,.\, \mathit{Tr}^{\phi}_F(\psi_1) \vee \ldots \mathit{Tr}^{\phi}_F(\psi_j))$. Suppose that $\mathbb{C}$ is any Π_2-realized skeleton and θ is a variable assignment.

If $\mathbb{C} \not\models_\theta \mathit{Tr}_F(\phi)$, then $\mathbb{C} \models_\theta \mathit{Tr}_F(\phi) \supset \mathit{Tr}^{\phi}_F(\psi_1) \vee \ldots \mathit{Tr}^{\phi}_F(\psi_j)$.

So suppose $\mathbb{C} \models_\theta \mathit{Tr}_F(\phi)$. By Lemma 5, Clause 4, $\mathsf{cp}(\mathit{Tr}_F(\phi)) = F(\mathbb{A}), \overline{\eta}$. By Def. 4, there exists $H \colon F(\mathbb{A}) \to \mathbb{C}$, and $\theta \restriction \overline{x} = (H \circ \overline{\eta}) \restriction \overline{x}$.

Case 1. Suppose that H is node-injective. By Thm. 1, there is a Π_1-realized $\mathbb{B}$ such that $\mathbb{A} \rightsquigarrow_1 * \, \mathbb{B}$ and, for some J, K, $F(\mathbb{A}) \xrightarrow{K}_{ni} F(\mathbb{B}) \xrightarrow{J}_{ni} \mathbb{C}$. By Lemma 3, Clause 2, $K = F(L)$ for some $L \colon \mathbb{A} \to \mathbb{B}$. Thus, $\mathbb{B} \models_{L \circ \eta} \phi$.

Since, by assumption, Π_1 achieves Γ, it follows that $\mathbb{B} \models_\zeta \psi_i$, for some ψ_i and some ζ s.t. $\zeta \restriction \overline{x} = (L \circ \eta) \restriction \overline{x}$. By Lemma 5, Clause 2, we can lift this to $F(\mathbb{B})$, so that $F(\mathbb{B}) \models_{\overline{\zeta}} \mathit{Tr}^{\phi}_F(\psi_i)$. Quantifying existentially, $F(\mathbb{B}) \models_{F(L) \circ \overline{\zeta}} \exists \overline{y} \,.\, \mathit{Tr}^{\phi}_F(\psi_i)$.

Applying J and using $K = F(L)$, we have $\mathbb{C} \models_{J\circ(K\circ\overline{\zeta})} \exists\overline{y}\,.\ Tr^{\phi}_{F}(\psi_i)$. Since $J \circ K = H$ and $\theta \restriction \overline{x} = (H \circ \overline{\zeta}) \restriction \overline{x}$, we have $\mathbb{C} \models_{\theta} \exists\overline{y}\,.\ Tr^{\phi}_{F}(\psi_i)$.

Case 2. H is not node-injective. By [17, Thm. 6.5], there is a universal K_0 among homomorphisms equating the nodes that H equates, and for some node-injective H_0, $H = H_0 \circ K_0$. Apply Case 1 to H_0. □