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 justifies inferring that a transformed protocol satisfies some security goals, when the source protocol did. Although a logical result about models of protocol behavior and the formulas they satisfy, it is a corollary of a logic-free theorem (Thm. 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 the protocol analysis activity into LTSs, and explaining relations between protocols using them, are new in this paper.

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

Structure of this paper. Section 2 introduces two protocols, with two transformations between them, motivating a definition of transformation (Def. 2). 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.

Strand Spaces. We work within the strand space theory [18]. A **strand** is the sequence of message transmissions and message receptions executed by a single principal in a single protocol session. Transmission and reception events jointly are **nodes**. We will write strands, either horizontally or vertically, as sequences of bullets connected by double arrows: $\bullet \Rightarrow \bullet$. (See [18, Sec. 2.4].)

A **protocol** Π consists of a finite set of strands, called the **roles** of the protocol, possibly annotated with some trust assumptions that we will not need here. We give each Π a "listener" role, consisting of a single node receiving a message t, which can witness for the disclosure of t. An **instance** of a role $\rho \in \Pi$ results from choosing values from some reasonable algebra $\mathfrak{M}$ of messages for ρ's parameters. We formalize this choice by applying a substitution α to ρ, where a substitution α means a homomorphism from $\mathfrak{M}$ to itself. The strand $\alpha(\rho)$ is the sequence of transmissions and receptions in which each message in ρ is instantiated according to α. That is, each parameter x appearing in ρ is replaced by $\alpha(x)$. Each strand $\alpha(\rho)$ is a **regular** behavior of some principal in Π, i.e. a local session in which the principal complies with Π. (See [18, Sec. 2.5].)

A fragmentary execution, called a **skeleton**, consists of a number of regular strands of Π, or their initial segments. A skeleton $\mathbb{A}$ consists of its regular nodes, equipped with (i) a partial ordering $\preceq_{\mathbb{A}}$, akin 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. (See [18, Def. 3.1].)

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 $\mathsf{non}(\mathbb{A})$. No value assumed freshly chosen should equal a guessed or independently chosen value. No long term key assumed uncompromised should be used by the adversary. (See [18, Defs. 3.2–4].) A realized skeleton is a full execution. When node n receives t, then the adversary obtains t, or can derive it using transmissions prior to n in the partial ordering $\preceq_{\mathbb{A}}$.

Any skeleton $\mathbb{A}$ represents a set of realized skeletons $\mathbb{B}$. These are the executions in which at least the transmissions and receptions in $\mathbb{A}$ have occurred, possibly made more specific by a substitution α, and node ordering extends $\preceq_{\mathbb{A}}$. Moreover, the assumptions $\alpha(\mathsf{unique}(\mathbb{A}))$ and $\alpha(\mathsf{non}(\mathbb{A}))$—i.e. the images of $\mathsf{unique}(\mathbb{A})$ and $\mathsf{non}(\mathbb{A})$ under α—must be satisfied.

Definition 1. The concatenation of two messages t_0 and t_1 is $t_0 \mathbin{\hat{}} t_1$, and the (asymmetric) encryption of t using the public encryption key of B is $\{\!|t|\!\}_{\mathsf{pk}(B)}$.

The i^{th} node along a strand s, starting from 1, is $s \downarrow i$.

We write $t_0 \sqsubseteq t_1$ to mean that message t_0 is an **ingredient** in t_1, i.e. 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 $t_0 \sqsubseteq \{\!|t_0|\!\}_K$; $t_0 \sqsubseteq t_0 \mathbin{\hat{}} t_1$, and $t_1 \sqsubseteq t_0 \mathbin{\hat{}} t_1$. 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, $N_b \sqsubseteq \{\!|N_b \mathbin{\hat{}} B|\!\}_{\mathsf{pk}(A)}$, but $\mathsf{pk}(A) \not\sqsubseteq \{\!|N_b \mathbin{\hat{}} B|\!\}_{\mathsf{pk}(A)}$.

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 at n when it was transmitted as an ingredient, but was neither transmitted nor received earlier on the same strand.

If $\rho \in \Pi$ is a role of protocol Π, then $\mathsf{instances}(\rho)$ is the set of instances of ρ, i.e. the set $\{\alpha(\rho) \colon \alpha \text{ is a substitution}\}$. The (larger) set of strands that agree with a member of $\mathsf{instances}(\rho)$ on the first i nodes is defined: $\mathsf{instances}(\rho|_i) = \{s \colon \exists r \in \mathsf{instances}(\rho) \,.\, \forall j \leq i \,.\, s \downarrow j = r \downarrow j\}$. So $\mathsf{instances}(\rho|_i)$ contains a strand if it is indistinguishable from a run of ρ while only i events have occurred. If $r \in \mathsf{instances}(\rho)$ agrees with s up to i, then r is an $\mathsf{instances}(\rho|_i)$-**witness for** s.

2 Some Protocol Transformations

The Protocol HD. HD, one of the simplest possible authentication protocols, is a half-duplex, authentication-only subprotocol of Needham-Schroeder [24]. It is shown in Fig. 1. HD gives the initiator an authentication guarantee that the responder has participated; it gives the responder no guarantee. HD does not establish any shared secret. The top half of the figure shows the initiator role, first transmitting the encrypted message for B and then receiving the freed nonce N. The bottom half shows the responder role, first receiving the encrypted message and then freeing and transmitting N'. A regular strand of HD is any instance of either one of these roles, using any values for the parameters N, B, N', B'. We do not show the listener role in this or other protocol diagrams, since it is always present, and simply passively receives a message t.

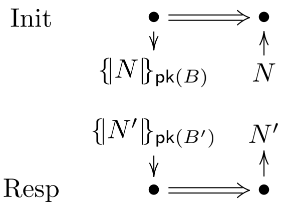
Init
$\{\!|N|\!\}_{\mathsf{pk}(B)}$
N
$\{\!|N'|\!\}_{\mathsf{pk}(B')}$
N'
Resp

Fig. 1. Protocol HD

A skeleton $\mathbb{A}$ over HD contains any number of these regular strands, and selects sets of freshness and non-compromise assumptions $\mathsf{unique}(\mathbb{A})$ and $\mathsf{non}(\mathbb{A})$.

$\mathbb{A}$ 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 assumptions $\mathsf{unique}(\mathbb{A})$ and $\mathsf{non}(\mathbb{A})$.

The Yes-or-No Protocol. 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. 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 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.

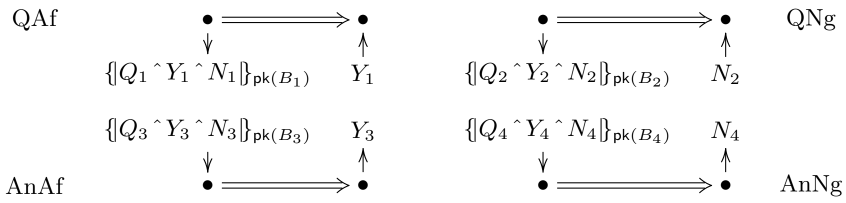
QAf
QNg
$\{\!|Q_1 \,\hat{}\, Y_1 \,\hat{}\, N_1|\!\}_{\mathsf{pk}(B_1)}$
Y_1
$\{\!|Q_2 \,\hat{}\, Y_2 \,\hat{}\, N_2|\!\}_{\mathsf{pk}(B_2)}$
N_2
$\{\!|Q_3 \,\hat{}\, Y_3 \,\hat{}\, N_3|\!\}_{\mathsf{pk}(B_3)}$
Y_3
$\{\!|Q_4 \,\hat{}\, Y_4 \,\hat{}\, N_4|\!\}_{\mathsf{pk}(B_4)}$
N_4
AnAf
AnNg

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

We can view either half of this diagram as a transformation of the protocol HD. We first rename the nonces N, N' of HD to the affirmative nonces Y_1, Y_3, to view the left half in this way. We would rename N, N' to the negative nonces N_2, N_4 for the right half. Before formalizing the transformations, we formalize these renamings as substitutions (homomorphisms) on the message algebra $\mathfrak{M}$. The renamings, yielding respectively the protocols $\alpha_1(\mathsf{HD})$ and $\alpha_2(\mathsf{HD})$, are:

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

We could alternatively incorporate substitutions such as α_1, α_2 into the transformations themselves, but at the cost of complicating the already subtle Def. 2.

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 Init to the Questioner's Affirmative role QAf. The first node of Init—the transmission node—is associated with the first node of QAf, and the second nodes are associated. F_1 sends Resp to the Answerer's Affirmative role AnAf, preserving node indices.
So $F_1(\text{Init}) = (\text{QAf}, \mathsf{Id})$ and $F_1(\text{Resp}) = (\text{AnAf}, \mathsf{Id})$.

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

These node index functions g are the identity Id, 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.

Our examples have several properties. The node index mapping functions are 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 $F_1(n)$ (see Def. 1). The reception node m of the HD responder receives $N \sqsubseteq \mathsf{msg}(m)$, and we also have $\alpha_1(N) \sqsubseteq \mathsf{msg}(F_1(m))$. 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 2 (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 $g(\mathsf{length}(\rho_1)) \leq \mathsf{length}(\rho_2)$;*
2. *$\rho_1 \downarrow i$ is a transmission (or resp. reception) node iff $\rho_2 \downarrow g(i)$ is;*
3. *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. *Suppose that ρ_1 and $\sigma_1 \in \Pi_1$ have a common instance up to i, i.e. $\alpha(\rho_1) \in \mathsf{instances}(\sigma_1|_i)$. Let $F(\sigma_1) = \sigma_2, h$. Then we have $g(j) = h(j)$ for all $j \leq i$. Moreover, $\alpha(\rho_2) \in \mathsf{instances}(\sigma_2|_{h(i)})$.*

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. 3)—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. A label ℓ indicates some unexplained behavior in $\mathbb{A}$ that prevents it from being realized; each $\mathbb{B}_i$ offers a different potential way to fix 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 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.

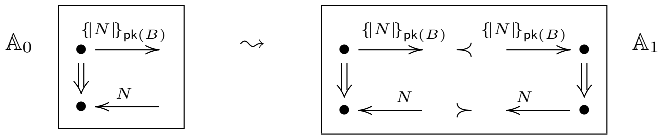
$\mathbb{A}_0$
$\{\!|N|\!\}_{\mathsf{pk}(B)}$
N
$\rightsquigarrow$
$\{\!|N|\!\}_{\mathsf{pk}(B)}$
$\prec$
$\{\!|N|\!\}_{\mathsf{pk}(B)}$
N
$\succ$
N
$\mathbb{A}_1$

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

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 $F(s)$, although we will not do so.

If we apply first α_1 and then F_1 mechanically to Fig. 3, we obtain the upper 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 $\mathbb{B}_2$.

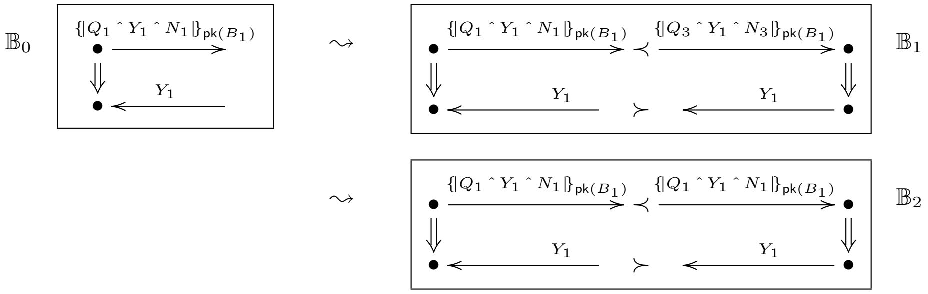
$\mathbb{B}_0$
$\{\!|Q_1 \mathbin{\hat{}} Y_1 \mathbin{\hat{}} N_1|\!\}_{\mathsf{pk}(B_1)}$
Y_1
$\rightsquigarrow$
$\{\!|Q_1 \mathbin{\hat{}} Y_1 \mathbin{\hat{}} N_1|\!\}_{\mathsf{pk}(B_1)}$
$\{\!|Q_3 \mathbin{\hat{}} Y_1 \mathbin{\hat{}} N_3|\!\}_{\mathsf{pk}(B_1)}$
$\mathbb{B}_1$
Y_1
Y_1
$\rightsquigarrow$
$\{\!|Q_1 \mathbin{\hat{}} Y_1 \mathbin{\hat{}} N_1|\!\}_{\mathsf{pk}(B_1)}$
$\{\!|Q_1 \mathbin{\hat{}} Y_1 \mathbin{\hat{}} N_1|\!\}_{\mathsf{pk}(B_1)}$
$\mathbb{B}_2$
Y_1
Y_1

Fig. 4. A goal met by YN, with $\mathsf{unique} = \{Y_1\}, \mathsf{non} = \{\mathsf{pk}(B_1)^{-1}\}$

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 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 [18, Def. 3.6]. A *homomorphism* $H\colon \mathbb{A} \longrightarrow \mathbb{B}$ 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 $\mathbb{A} \overset{H}{\longmapsto} \mathbb{B} \overset{J}{\longmapsto} \mathbb{A}$ does not imply that $J \circ H$ is an isomorphism. However, if H, J map distinct nodes of their sources injectively to distinct nodes of their targets, then $J \circ H$ is an isomorphism (under reasonable assumptions about the algebra of messages). Thus, these *node-injective* homomorphisms $H \colon \mathbb{A} \longmapsto_{ni} \mathbb{B}$ determine a partial order $\leq_{ni}$ on skeletons to within isomorphism.

Lemma 1 ([18, 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 $\longmapsto$. Protocol analysis is a search through part of the preorder $\longmapsto$. Skeletons $\mathbb{A}_0$ determine starting points for the search; protocol analysis then seeks realized skeletons $\mathbb{C}$ such that $\mathbb{A} \longmapsto \mathbb{C}$. CPSA computes a set of representative realized skeletons we call *shapes.* Within the set of all realized $\mathbb{B}$ such that $\mathbb{A} \longmapsto \mathbb{B}$, the shapes are the *minimal* ones in the node-injective ordering $\leq_{ni}$ [13]. 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 $\longmapsto$.

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 [18, 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* Δ mapping Π_1 labels to Π_2 labels. A simulation using Δ *progresses* iff, whenever a Π_1 skeleton $\mathbb{A}$ can take some ℓ-transition, the Π_2 skeleton $F(\mathbb{A})$ can take some $\Delta(\ell)$ transition. If for some Δ, 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 3. *Let S be a set of skeletons, and* 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: there exists a $\mathbb{B}$ such that $\mathbb{A} \rightsquigarrow \mathbb{B}$ iff $\mathbb{A}$ is not realized;*
2. *If $\mathbb{A} \overset{\ell}{\rightsquigarrow} \mathbb{B}$, then:*
   (a) If $\ell =$ dead*, then $\mathbb{A} = \mathbb{B}$ and there is no realized $\mathbb{C}$ such that $\mathbb{A} \longmapsto \mathbb{C}$;*
   (b) If $\ell \neq$ 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* $\mathbb{A} \xrightarrow{H} \mathbb{B}' \xrightarrow{K} \mathbb{C}$.

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, where c is a value received in a new, unexplained form in the node n_1. "Unexplained" because in n_1 c is not contained within a set of encryptions E. We call E an "escape set," since c has escaped from the protection of the encryptions E before reaching 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. 2). 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 ([17]). *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 $\Delta\colon \Lambda(\Pi_1) \to \Lambda(\Pi_2)$. The $\mathsf{Skel}(\Pi_1)$ argument determines what parameters in $\mathbb{A}$ to avoid, when choosing new role parameters.

Definition 4. *1. F, Δ* preserve progress *for $\rightsquigarrow_1$ and $\rightsquigarrow_2$ iff: (a) $\ell = \mathsf{dead}$ iff $\Delta(\ell) = \mathsf{dead}$, and (b) for every $\ell \in \Lambda$, $\mathbb{A} \overset{\ell}{\rightsquigarrow}_1$ implies $F(\mathbb{A}) \overset{\Delta(\ell)}{\rightsquigarrow}_2$.*

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

If F, Δ 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 $\mathrm{ran}(\Delta)$, for instance the second step in Fig. 4. In F_1, F_2, Δ is determined directly from F_i; in other cases, $\Delta(\ell)$ 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 $\Delta\colon \Lambda(\Pi_1) \to \Lambda(\Pi_2)$. 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, Δ preserve progress for $\rightsquigarrow_1$ and $\rightsquigarrow_2$;
2. $\rightsquigarrow_1$ simulates $\rightsquigarrow_2$ under F, Δ.

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

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

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. 3, 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 $\ell \neq \mathsf{dead}$, $\mathbb{A} \overset{\ell}{\rightsquigarrow}_1$. By progress, $F(\mathbb{A}) \overset{\Delta(\ell)}{\rightsquigarrow}_2$. By Def. 3, Cl. 2c, H factors through some member of $\{\mathbb{E}\colon F(\mathbb{A}) \overset{\Delta(\ell)}{\rightsquigarrow}_2 \mathbb{E}\}$, say $\mathbb{E}_0$. By simulation, $\mathbb{E}_0 = F(\mathbb{A}')$ for some $\mathbb{A}'$ with $\mathbb{A} \overset{\ell}{\rightsquigarrow}_1 \mathbb{A}'$. By Defn. 3, 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, Δ 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 [16].[1] A formula

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

[1] Although the syntax is simplified from [16], $\mathcal{L}(\Pi)$'s expressiveness is unchanged.

is a *security goal* if (i) ϕ and each ψ_i is a conjunction of atomic formulas, (ii) $\overline{x}$ and $\overline{y}$ are disjoint lists of variables, and (iii) all variables free in any ψ_i but not in ϕ is free in $\overline{y}$. Null and unary disjunctions ($j = 0$ or $j = 1$) are permitted, where the null disjunction $\perp$ is the constantly false formula.

Since $\mathcal{L}(\Pi)$ says nothing about the structure of Π's messages, it can express goals that are preserved when 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 $\texttt{pk}(a)$, $\texttt{sk}(a)$, and $\texttt{inv}(a)$, for a's public encryption key; a's private signature key; and the inverse member of a key pair.

Example 1: $\mathcal{L}$(HD). $\mathcal{L}$(HD) shares some vocabulary with the other $\mathcal{L}(\Pi)$:

$$\texttt{Prec}(m,n) \qquad \texttt{Unq}(v) \qquad \texttt{UnqAt}(n,v) \qquad \texttt{Non}(v)$$

expressing node precedence; unique origination of v; unique origination of v at the node n; and non-origination of v. Its protocol-specific relations are:

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

The predicates AtPos1 and AtPos2 say that a node n lies at the first or second position (resp.) on strand s. Init and Resp say that a strand s is an initiator or responder strand (resp.). $\texttt{Lsn}(s)$ says that s is a listener strand, a purely receptive strand that can witness that a value is disclosed. Peer says that s's name parameter is b, if s is an initiator strand. This name refers to s's intended partner. Self says that s's name parameter is b, if s is a responder strand. This name refers to the active party in this strand, the owner of the decryption key. We use Nonce to say that v is the nonce in either type of strand s. Hear says that s has received the value v, if s is a listener strand. The skeleton $\mathbb{A}_0$ of Fig. 3 *satisfies* the formula $\Phi_0 =$

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

when we assign s to the initiator strand shown; assign n and m to its first and second nodes resp.; assign b to the name B; and assign v to nonce N. If $\Phi_1 =$

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

then $\mathbb{A}_1$ satisfies $\Phi_0 \wedge \Phi_1$, when we (additionally) assign s' to the responder strand shown, and 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. So $\Phi_0 \supset \Phi_1$ is an authentication goal.

Although HD does not achieve any confidentiality goal, we would express it with a null conclusion, and with the Lsn and Hear predicates in the hypothesis. Letting $\perp$ be the vacuously false null disjunction:

$$\begin{array}{c} (\ \texttt{Init}(s) \ \wedge \ \ \texttt{AtPos1}(s,n) \ \ \wedge \ \texttt{Peer}(s,b) \\ \texttt{Nonce}(s,v) \wedge \texttt{Non}(\texttt{inv}(\texttt{pk}(b))) \wedge \texttt{UnqAt}(n,v) \\ \texttt{Lsn}(s') \ \ \wedge \ \texttt{AtPos1}(s',n') \ \wedge \texttt{Hear}(n',b)\ ) \quad \supset \perp \end{array}$$

would express the secrecy of b.

$\mathcal{L}(\Pi)$ in General. In the previous paragraphs, we created an intuitive relation between a particular protocol Π and the predicates of its language. To do the same thing more generally, we will use a table τ.

The table τ should associate a distinct one place predicate with each role. For HD, τ mapped the initiator role to the $\mathtt{Init}$ predicate and the responder role to $\mathtt{Resp}$. These are the *role predicates.* We also have τ associate each pair of a role and parameter of that role to a two place predicate. For HD, τ mapped the initiator role and the parameter N to the predicate $\mathtt{Nonce}$; it also maps the responder role and N' to $\mathtt{Nonce}$. These are the *parameter predicates.* We require (i) role predicates and parameter predicates are disjoint, and (ii) τ maps different parameters of the same role to different parameter predicates. For precision, we could write $\mathcal{L}_\tau(\Pi)$ for the language determined by a particular choice of τ, but the choice of τ makes no semantically relevant difference in security goals.

If the longest role in Π is of length k, we add k position predicates $\mathtt{AtPos1}(s,n)$, ..., $\mathtt{AtPos}k(s,n)$. $\mathcal{L}(\Pi)$ contains these position predicates, the shared predicates $\mathtt{Prec}, \mathtt{Unq}, \mathtt{UnqAt}, \mathtt{Non}$, and the predicates in the range of τ.

Classical Semantics of $\mathcal{L}(\Pi)$. Fix a message algebra $\mathfrak{M}$. Each skeleton $\mathbb{A}$ for Π determines a model for $\mathcal{L}(\Pi)$ where the domain contains the messages in $\mathfrak{M}$ together with the strands and nodes of $\mathbb{A}$. Given an assignment η mapping free variables to values in the domain, the value $\eta(v)$ of any term v, involving variables and functions symbols such as $\mathtt{pk}$, etc., is determined by η and $\mathfrak{M}$. The satisfaction conditions for the predicates are:

$\mathbb{A} \models_\eta \mathtt{Prec}(n,n')$ iff $\eta(n) \prec_{\mathbb{A}} \eta(n')$;
$\mathbb{A} \models_\eta \mathtt{Non}(v)$ iff $\eta(v) \in \mathsf{non}_{\mathbb{A}}$;
$\mathbb{A} \models_\eta \mathtt{Uniq}(v)$ iff $\eta(v) \in \mathsf{unique}_{\mathbb{A}}$;
$\mathbb{A} \models_\eta \mathtt{UniqAt}(n,a)$ iff $\eta(v) \in \mathsf{unique}_{\mathbb{A}}$ and $\eta(v)$ originates at $\eta(n)$ in $\mathbb{A}$;
$\mathbb{A} \models_\eta \mathtt{AtPos}k(s,n)$ iff $\eta(n) \in \mathsf{nodes}_{\mathbb{A}}$ and $\eta(n) = \eta(s) \downarrow k$;
$\mathbb{A} \models_\eta \mathtt{RolePred}(s)$ iff for some $j > 0$, $\mathsf{nodes}_{\mathbb{A}}$ contains exactly j nodes of $\eta(s)$, and $\eta(s) \in \mathsf{instances}(\rho|_j)$, where $\tau(\rho)$ is $\mathtt{RolePred}$;
$\mathbb{A} \models_\eta \mathtt{ParamPred}(s,v)$ iff for some $j > 0$, $\mathsf{nodes}_{\mathbb{A}}$ contains exactly j nodes of $\eta(s)$, and $\eta(s) \in \mathsf{instances}(\rho|_j)$, and for every $\mathsf{instances}(\rho|_j)$-witness $\alpha(\rho)$ for $\eta(s)$ has $\alpha(a) = \eta(v)$, where $\tau(\rho, a)$ is $\mathtt{ParamPred}$.

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. 2, Clause 5, $\mathcal{L}(\Pi)$ is insensitive to not-yet-executed branch points.

This definition also justifies our claims about Φ_0 and $\Phi_0 \wedge \Phi_1$ above. 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 $H\colon \mathbb{A}_0 \mapsto \mathbb{C}$, 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 $H\colon \mathbb{A}_0 \mapsto \mathbb{C}$, $\theta = H \circ \eta$.

Definition 5. *$\mathbb{A}, \eta$ is a Φ-*characteristic pair *iff (i) $\mathbb{A} \models_\eta \Phi$ and (ii), for all $\mathbb{B}, \theta$, if $\mathbb{B} \models_\theta \Phi$ implies $\exists! H\,.\ H\colon \mathbb{A} \mapsto \mathbb{B}$ and for all $x \in \mathsf{fv}(\Phi)$, $\theta(x) = (H \circ \eta)(x)$.*

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

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 $\Gamma_1 =$

$$\forall s, n, m, b, v \,.\; (\Phi_0 \quad \supset \quad \exists s', n', m' \,.\; \Phi_1). \tag{2}$$

Role-Specific Formulas. 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 *role specific* 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 role specific, but Φ_1 alone is not. Node variables m, n occur in no position predicate in Φ_1. By associativity and commutativity, a role specific ϕ can be rewritten so every leftmost subformula is role specific. That is, ϕ is equivalent to $\phi_0 \wedge \psi_1$ where ϕ_0 is role specific, and if ϕ_0 is a conjunction $\phi_1 \wedge \psi_1$, then ϕ_1 satisfies the same property recursively. This holds because 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 [16, Thm. 5.2]:

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

If Γ is a goal formula as in Eqn. (1), then we say that Γ is role specific 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 `AtPos1`(s, n) and `AtPos2`(s, n). It has five role predicates, namely `QAf`(s), `QNg`(s), `AnAf`(s), `AnNg`(s), and `Lsn`(s). Again expressing an intended peer via `Peer`(s, b), and an actual identity via `Self`(s, b), we have five parameter predicates:

`Quest`(s, q), `YesVal`(s, v), `NoVal`(s, v), `Peer`(s, b), `Self`(s, b).

Protocol Transformations and Language Translations. Each $F \colon \Pi_1 \rightarrow \Pi_2$ determines a translation $Tr_F(\cdot)$ between role specific goal formulas of $\mathcal{L}(\Pi_1)$ and $\mathcal{L}(\Pi_2)$. We translate conjunctions by translating each conjunct. Let $F(\rho_1) = (\rho_2, g)$. The order and assumption predicates are translated verbatim. For the remaining predicates, we use the tables τ_1 and τ_2.

`RolePred1`(s)**:** If `RolePred1` is $\tau_1(\rho_1)$, translate it to $\tau_2(\rho_2)$.

`PosPred`$i(s, n)$**:** Because the formula is role specific, there is a single conjunct `RolePred1`(s) with the same s. If `RolePred1` is $\tau_1(\rho_1)$, then letting the index $j = g(i)$, the result is `PosPred`$j(s, n)$.

$\mathtt{ParamName1}(s,t)$**:** Again, there is a single conjunct $\mathtt{RolePred1}(s)$ with the same s, by role specificity. Suppose $\mathtt{RolePred1}$ is $\tau_1(\rho_1)$, and $\mathtt{ParamName1}$ is $\tau_1(\rho_1, a)$. Select the predicate $\mathtt{ParamName2}$ to be $\tau_2(\rho_2, a)$.
If either (i) a does not appear in ρ_2, or (ii) for all parameters a of ρ_1, $\mathtt{ParamName1}$ is not $\tau_1(\rho_1, a)$, then the result is vacuously true: $s = s \wedge t = t$.

If $\phi \wedge \psi_i$ is role specific, $\mathit{Tr}^{\phi}_F(\psi)$ translates ψ_i the same way, using conjuncts of ϕ to provide the specification of the roles. We have ensured that $\mathsf{fv}(\mathit{Tr}_F(\phi)) = \mathsf{fv}(\phi)$ and $\mathsf{fv}(\mathit{Tr}^{\phi}_F(\psi)) = \mathsf{fv}(\psi)$. So, let $\mathit{Tr}_F(\forall \overline{x}\,.\,(\phi\;\supset\;\exists\overline{y}\,.\,\psi_1 \vee \ldots \vee \psi_k))$ be

$$\forall \overline{x}\,.\,(\mathit{Tr}_F(\phi)\;\supset\;\exists \overline{y}\,.\;\mathit{Tr}^{\phi}_F(\psi_1) \vee \ldots \vee \mathit{Tr}^{\phi}_F(\psi_k)). \tag{3}$$

For the goal formula $\mathit{\Gamma}_1$ describing Fig. 3, $\mathit{Tr}_{F_1}(\mathit{\Gamma}_1)$ is:

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

If η is an variable assignment taking values in $\mathbb{A}$, define $\overline{\eta}$ to be the corresponding assignment taking values in $F(\mathbb{A})$. In particular, $\overline{\eta}$ agrees with η for variables whose value is a message. For variables whose value is a strand or node, we have $\overline{\eta}(v) = F(\eta(v))$, 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 $F(\mathbb{A}) \models_{\overline{\eta}} \mathit{Tr}_F(\phi)$.*
2. If $\mathbb{A} \models_{\eta} \phi \wedge \psi$, then $F(\mathbb{A}) \models_{\overline{\eta}} \mathit{Tr}^{\phi}_F(\psi)$.
3. If $\mathbb{B} \models_{\theta} \mathit{Tr}_F(\phi)$ 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. If ϕ is role specific, then $\mathsf{cs}(\mathit{Tr}_F(\phi))$ exists, and $F(\mathsf{cs}(\phi)) = \mathsf{cs}(\mathit{Tr}_F(\phi))$.

Proof. **1.** We left-associate conjunctions, so that when ϕ is $\phi_1 \wedge \phi_2$, ϕ_2 is atomic. We also assume that ϕ is written in the order defined before Lemma 4, so every leftmost subformula of ϕ is also role specific. We use structural induction on ϕ.
Base case: If ϕ is the trivially true conjunction with 0 conjuncts, then $\mathit{Tr}_F(\phi)$ is also the null conjunction, which is true in every structure.
Induction step: Let ϕ be $\phi_1 \wedge \phi_2$, where the claim holds for ϕ_1, and ϕ_2 is atomic. When ϕ_2 uses $\mathtt{Prec}, \mathtt{Unq}, \mathtt{UnqAt}, \mathtt{Non}$, or $=$, the claim follows from the definitions.

If ϕ_2 is a role predicate $\mathtt{RoleName}(s)$, $\eta(s)$ 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 $\mathtt{ParamName}(s,t)$. Since ϕ is role specific, there is just one $\mathtt{RoleName}(s)$ with the same s in ϕ_1, and $\mathbb{A} \models_{\eta} \mathtt{RoleName}(s)$. 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, $F(\mathbb{A}) \models_{\overline{\eta}} \mathit{Tr}_F(\phi_2)$.

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 [16], 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 $J\colon F(\mathsf{cs}(\phi)) \longmapsto \mathbb{B}$. $J = [\phi, \alpha]$ 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 role specific, so Lemma 4 implies $\mathsf{cs}(\mathit{Tr}_F(\phi))$ exists. By the previous clause, $J\colon F(\mathsf{cs}(\phi)) \longmapsto \mathsf{cs}(\mathit{Tr}_F(\phi))$.

By clause 1, $F(\mathsf{cs}(\phi)) \models_{\overline{\eta}} \mathit{Tr}_F(\phi)$. Thus, Def. 5 entails that $K\colon \mathsf{cs}(\mathit{Tr}_F(\phi)) \longmapsto F(\mathsf{cs}(\phi))$. Hence, by the uniqueness in Def. 5, $J \circ K = \mathsf{Id}$. Hence, $F(\mathsf{cs}(\phi))$ and $\mathsf{cs}(\mathit{Tr}_F(\phi))$ are isomorphic. □

7 Preserving Security Goal Formulas

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

Theorem 2 *Let $F\colon \mathit{\Pi}_1 \to \mathit{\Pi}_2$, and let ϕ be goal specific with $\mathbb{A} = \mathsf{cs}(\phi)$.*

Suppose $\mathit{\Pi}_1$ achieves the goal $\mathit{\Gamma} = \forall \overline{x}\,.\,(\phi \supset \exists \overline{y}\,.\,\psi_1 \vee \ldots \psi_j)$, and assume there exists a $\mathit{\Delta}$ and TLTS*s $\rightsquigarrow_1$ and $\rightsquigarrow_2$ as in Thm. 1, i.e. $\mathbb{A} \in S(\rightsquigarrow_1)$ and*

1. *$F, \mathit{\Delta}$ preserve progress for $\rightsquigarrow_1$ and $\rightsquigarrow_2$;*
2. *$\rightsquigarrow_1$ simulates $\rightsquigarrow_2$ under $F, \mathit{\Delta}$.*

Then $\mathit{\Pi}_2$ achieves $\mathit{Tr}_F(\mathit{\Gamma})$.

Proof. $\mathit{Tr}_F(\mathit{\Gamma})$ 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 $\mathit{\Pi}_2$-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. 5, there exists $H\colon F(\mathbb{A}) \longmapsto \mathbb{C}$, 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 $\mathit{\Pi}_1$-realized $\mathbb{B}$ such that $\mathbb{A} \rightsquigarrow_1 * \,\mathbb{B}$ and, for some J, K, $F(\mathbb{A}) \overset{K}{\longmapsto}_{ni} F(\mathbb{B}) \overset{J}{\longmapsto}_{ni} \mathbb{C}$. By Lemma 3, Clause 2, $K = F(L)$ for some $L\colon \mathbb{A} \longmapsto \mathbb{B}$. Thus, $\mathbb{B} \models_{L \circ \eta} \phi$.

Since, by assumption, $\mathit{\Pi}_1$ achieves $\mathit{\Gamma}$, 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 $\mathbb{C} \models_{J \circ (K \circ \overline{\zeta})} \exists \overline{y}\,.\,\mathit{Tr}^{\phi}_F(\psi_i)$. Since $J \circ K = H$ and $\theta \restriction \overline{x} = (H \circ \overline{\zeta}) \restriction \overline{x}$, we have $\mathbb{C} \models_{\theta} \exists \overline{y}\,.\,\mathit{Tr}^{\phi}_F(\psi_i)$.

Case 2. H is not node-injective. By [18, Thm. 6.5], there is a universal $K_0\colon F(\mathbb{A}) \longmapsto \mathbb{D}$ 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 and $\mathbb{D}$. □

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. [5]. In a key special case, "protocol composition," it dates from the 1990s [21, 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 TLTS 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 [25, 3] or universal composability [7]; weaker forms may still be cryptographically justified [12].

In the *symbolic* model, we previously provided a widely applicable and practically useful criterion [19, 14]. Cortier et al.'s criterion is in some ways broader but in other ways narrower than ours [8]; cf. [1]. Our [15] covers the union of [19, 8, 1]. From one point of view, the contribution of the present paper is to generalize [15] beyond the composition case.

The Protocol Composition Logic PCL considers refinements that preserve security goals [11, 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. [11]'s "parallel" and "sequential" composition amounts to $\Pi_1 \subseteq \Pi_2$. Datta et al.'s "protocol refinement using templates" [10] suggested many of our examples.

By contrast with Distributed Temporal Logic [6], $\mathcal{L}(\Pi)$ is intended to be less expressive about the forms of messages. We wanted to focus only on what is retained under transformation, which concerns the role parameters rather than the forms of the messages. Nevertheless, our logic, unlike DTL, being a quantified logic, satisfaction is undecidable.

Lowe and Auty [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 [9, 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 [19,15], cf. [23,8]. 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, Dusko Pavlovic, John Ramsdell, Paul Rowe, and Javier Thayer. The simplification of $\mathcal{L}(\Pi)$ vs. [16] arose from a conversation with Ramsdell. Early versions of some of this material were presented at FCS-ARPSA-WITS in 2008 and in Darmstadt in 2010.

Thanks to Siraj Sayani and Soumentra Ghosal, whose hospitality I enjoyed in Coonoor while writing a 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. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.

11. 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.
12. 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.
13. 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.
14. Joshua D. Guttman. Authentication tests and disjoint encryption: a design method for security protocols. *Journal of Computer Security*, 12(3/4):409–433, 2004.
15. 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.
16. Joshua D. Guttman. Security theorems via model theory. *EXPRESS: Expressiveness in Concurrency (EPTCS)*, 8:51, 2009. doi:10.4204/EPTCS.8.5.
17. 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.
18. 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.
19. Joshua D. Guttman and F. Javier Thayer. Protocol independence through disjoint encryption. In *Computer Security Foundations Workshop*. IEEE CS Press, 2000.
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.