Searching for Shapes in Cryptographic Protocols*

Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer

The MITRE Corporation
shaddin, guttman, jt@mitre.org

Abstract. A *shape* describes the behavior of the honest participants in some minimal protocol execution. These shapes are informative, because typically protocols have very few of them. Authentication and secrecy properties are easy to determine from the set of shapes, as are attacks, and other protocol characteristics.
A *skeleton* gives partial information about some possible executions, and a *homomorphism* from one skeleton to another is an information-preserving map. We describe a procedure that searches through skeletons using homomorphisms. The search procedure has been implemented in a Cryptographic Protocol Shape Analyzer CPSA.

1 Introduction

The executions of cryptographic protocols frequently have a limited number of essentially different possible forms, which we call shapes. By enumerating these shapes, we may ascertain whether they all satisfy a security condition such as an authentication or confidentiality property. Enumerating the shapes of a protocol also allows us to find out whether any execution has other anomalies, which are not necessarily counterexamples to the security goals, such as involving unexpected participants, or involving more local runs than expected.

In this paper, we describe a method for enumerating the shapes that a protocol admits. If a protocol permits a security failure or an anomaly of other kinds, then the shapes found in the enumeration will show it. If the protocol has only finitely many essentially different shapes, the enumeration will terminate, allowing us to conclude that no anomaly exists. We can then read off the answers to traditional secrecy and authentication questions from the shapes.

We use the strand space theory as a framework [9]. A *skeleton* [4, 5] represents regular (non-penetrator) behavior that might make up part of an execution, and a *homomorphism* is an information-preserving map between skeletons. A skeleton is *realized* if it contains exactly the regular behavior of some execution. A realized skeleton is a *shape* if it is minimal in the sense of Definition 11. We *search* for shapes using the authentication tests [9] to find new strands to

* Supported by the National Security Agency and by MITRE-Sponsored Research.

add when a skeleton is not large enough to be realized. The shape enumeration reflects protocol executions of all sizes. Our work does not lie within the widely practiced *bounded* protocol analysis (initiated in [2, 7, 14]).

Our main result is that the search is *complete*, in the sense that—for any protocol—our search discovers every shape for that protocol. It cannot terminate for every protocol [6], but it does apparently terminate for reasonably inclusive classes [1, 17]. The search idea is related to the second version of Athena [16], which adopted the authentication tests from early versions of [9]. However, our search involves the regular behaviors alone; we never represent penetrator activity within a shape. Moreover, we have improved both the search and the theory; we have introduced the notion of shape; and we have created versions of the authentication tests strong enough for completeness to be true.

We implemented this search in a program called the Cryptographic Protocol Shape Analyzer (CPSA).

Road map to this paper. Section 2 describes examples, including the Yahalom protocol, a small but interesting case for our method. An intuitive analysis leads to a single shape. The remainder of the paper formalizes this kind of argument.

Section 3 summarizes definitions, and Section 4 introduces skeletons, homomorphisms, and shapes. Section 5, the heart of the paper, extracts search steps from the authentication tests [9]. These search steps are complete (Corollary 3).

Section 6 describes the search algorithm, which we illustrate with a rigorous treatment of Yahalom's protocol (Section 7). In Section 8, we describe the implementation of the search method in CPSA.

2 Some Examples

In practice, protocols have remarkably few shapes. The Needham-Schoeder-Lowe protocol has one, shown in Fig. 1, where we use $\{\!|t|\!\}_K$ to refer to the encryption

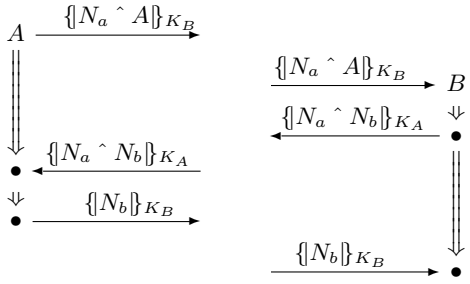

Fig. 1. Only Shape for Needham-Schroeder-Lowe

of t with key K, and $t \hat{} t'$ to refer to the concatenation of messages. In this

analysis, we take B's point of view, assuming that a full local run of B's side has occurred. If B's nonce N_b has been freshly chosen and A's private key K_A^{-1} is uncompromised, then A must have had a matching run, with the messages sent and received in the order shown (reading down the page). The same diagram also represents A's point of view, if we instead assume both private keys K_A^{-1}, K_B^{-1} uncompromised and A's nonce N_a freshly chosen. However, the lowermost node of B's run may be absent.

Uniqueness of shape is not surprising for such a strong protocol. However, an incorrect protocol such as Needham-Schroeder may also have a single shape, as shown (from B's point of view) in Fig. 2. We again assume that B has executed a full run, using a fresh nonce N_b, and that A's private key K_A^{-1} is uncompromised. Although there is a single shape, there are two ways that this shape may be realized. Either (1) C's private key may be compromised, in which case we may complete this diagram with adversary activity to obtain the Lowe attack [12]; or else (2) $C = B$, leading to the intended run.

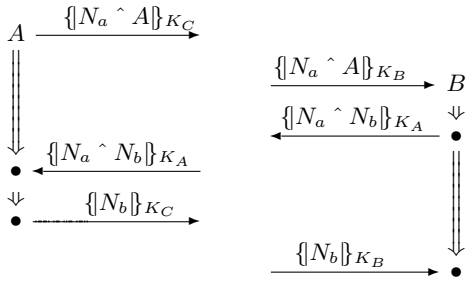

Fig. 2. Only Shape for Needham-Schroeder

Some protocols have more than one shape, Otway-Rees, e.g., having four. One is the intended shape. The others are variants of one basic situation, in which the responder B "thinks he wants to talk to himself," in the sense that the intended initiator is also B. There need not be any initiator run at all. This anomaly was first observed by Michael Goldsmith using Casper and FDR [13].

In searching for the shapes of a protocol, one always starts from some initial behavior. Typically, it consists of one single local run; we refer to this original run as the "point of view" of the analysis.

2.1 The Yahalom Protocol

The Yahalom protocol [3] provides a session key K to principals sharing long-term symmetric keys with a key server S. We let $\mathsf{ltk}(\cdot)$ map each principal A to its long term shared key $\mathsf{ltk}(A)$.

The protocol contains three roles, the initiator, the responder, and the server. Each is described by one column in Fig. 3, and each role is parametrized by

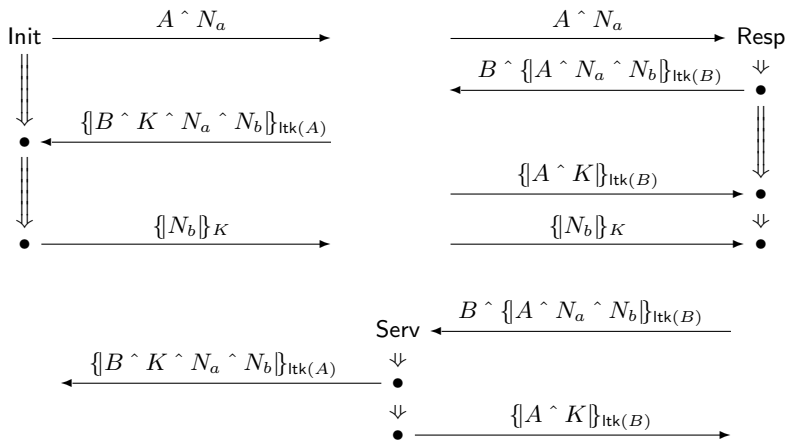

Fig. 3. Yahalom protocol (forwarding removed)

A, B, N_a, N_b, K. The behavior Init of the initiator consists transmitting $A \hat{} N_a$ followed by receiving $\{|B \hat{} K \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(A)}$ and finally transmitting $\{|N_b|\}_K$; instances of this role are obtained by plugging in values for the parameters. The other roles are equally self-explanatory. The key server is trusted to generate a fresh session key K in each run, so we will assume it to originate nowhere else.

2.2 Yahalom Shapes for the Responder

Suppose an execution contains a local run of the responder's role as shown in the upper right column of Fig. 3. What shapes of global run are possible?

We assume the long term keys $\mathsf{ltk}(A), \mathsf{ltk}(B)$ of the two participants are uncompromised. Similarly, we assume the responder's nonce N_b to be fresh and unguessable.

Step 1: A server run. N_b is transmitted in B's second step (n_0 in Fig. 4) within the encrypted unit $\{|A \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(B)}$. In B's fourth step (n_1 in Fig. 4), it is received outside that unit in the form $\{|N_b|\}_K$.

Since N_b was freshly chosen and unguessable, it must be extracted from B's message, and not obtained from any independent activity. Since $\mathsf{ltk}(B)$ is uncompromised, the adversary would never be able to get it out of its original message. Since some participant does get it out of this form, that must be an uncompromised participant, namely some protocol participant willing to receive a message containing $\{|A \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(B)}$ and later retransmit N_b outside this unit. The structure of the protocol in Fig. 3, provides only one possibility: a run of the server role. The parameters appearing in $\{|A \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(B)}$ ensure that the server uses the parameters A, B, N_a, N_b. This server run generates some session key K'. We do not yet know whether $K' = K$. The situation so far is described in Fig. 4.

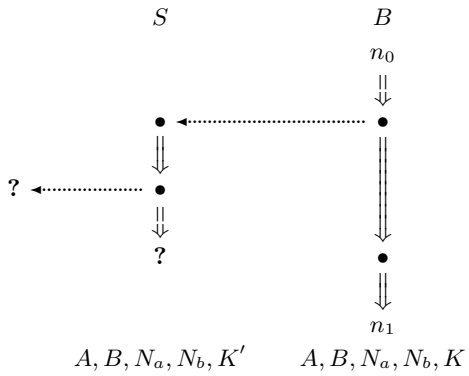

Fig. 4. Adding a server run

The authentication test idea. We will repeatedly use the form of reasoning we just used in Step 1. We call it an authentication test. The nodes at which N_b is freshly transmitted and later received back form a *test*, from which we can infer an action taken by a protocol participant. Since the fresh value goes out in an encrypted message from which it can be extracted only using an uncompromised key, only a participant using this key according to the protocol can extract N_b and retransmit it in a new form. We can take cases on the transformations the protocol offers to see how the extraction could have occurred. Indeed, often there is no choice: only one protocol action could achieve the transformation.

In this context, we will speak of the *test nodes* that require a transformation; in Step 1, the test nodes are n_0, n_1. The new protocol activity (the server behavior in Step 1) provides a *transforming edge*. In Step 1, the test nodes require the transforming edge to extract N_b from a single encrypted unit. However, in other cases, for instance the one we turn to next, the transforming edge may need to extract it from several different forms.

Step 2: An initiator run. N_b was sent at n_0 inside $\{\!|A \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(B)}$, and the server retransmits it inside $\{\!|B \hat{} K' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}$. However, it eventually reaches B at n_1 outside these encrypted messages as $\{\!|N_b|\!\}_K$.

With $\mathsf{ltk}(A), \mathsf{ltk}(B)$ uncompromised, the adversary cannot extract N_b from the set of forms $\{\, \{\!|A \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(B)}, \{\!|B \hat{} K' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)} \,\}$. Thus, we have an authentication test, and some regular participant must free N_b from these encrypted units. The structure of the protocol in Fig. 3, provides only one possibility: a run of the initiator role. The parameters appearing in $\{\!|B \hat{} K' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}$ ensure that the initiator is A, running with the parameters A, B, N_a, N_b, K'. (See Fig. 5.)

Step 3: Does $K = K'$***?*** Assuming that $K' \neq K$, we may use another authentication test. N_b was transmitted only in the forms $\{\!|A \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(B)}$, $\{\!|B \hat{} K' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}$, and lastly $\{\!|N_b|\!\}_{K'}$. However, the test nodes n_0, n_1

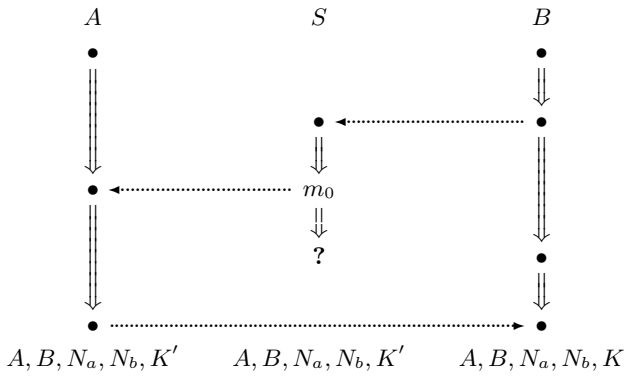

Fig. 5. Adding an initiator run

show that it is extracted from all of these encrypted units, to be received as $\{\!|N_b|\!\}_K$. If we assume for the moment that K' remains uncompromised, like $\mathsf{ltk}(A)$ and $\mathsf{ltk}(B)$, we can again apply the authentication test idea.

However, the protocol in Fig. 3 provides no transforming edges at all: a participant following the protocol that receives N_b only in the forms

1. $\{\!|A \mathbin{\hat{}} N_a \mathbin{\hat{}} N_b|\!\}_{\mathsf{ltk}(B)}$,
2. $\{\!|B \mathbin{\hat{}} K' \mathbin{\hat{}} N_a \mathbin{\hat{}} N_b|\!\}_{\mathsf{ltk}(A)}$ for $K' \neq K$, and
3. $\{\!|N_b|\!\}_{K'}$ for $K' \neq K$

does not retransmit N_b in any other form. So the assumption $K' \neq K$ is false.

Step 4: K' ***is uncompromised.*** We used the premise that K' is uncompromised, which we will now check. It depends on our trust that the key server, in any run, will always select a fresh and unguessable session key K'. In Fig. 5, it is transmitted first at node m_0. If it is ever received in compromised form at any node m_1, then there must be a transforming edge that extracts K' from the encrypted units in which the server transmits K', namely $\{\!|B \mathbin{\hat{}} K' \mathbin{\hat{}} N_a \mathbin{\hat{}} N_b|\!\}_{\mathsf{ltk}(A)}$ and $\{\!|A \mathbin{\hat{}} K'|\!\}_{\mathsf{ltk}(B)}$.

However, the protocol in Fig. 3 provides no possibilities: no participant ever receives a key K' and retransmits it as part of the plaintext of a different message. So it is impossible for K' to become compromised.

Step 5: B ***receives*** K ***from the same server run.*** Since B receives the message $\{\!|A \mathbin{\hat{}} K|\!\}_{\mathsf{ltk}(B)}$, this message must have been transmitted. Since $\mathsf{ltk}(B)$ is uncompromised, the adversary cannot prepare it. Therefore it must have been transmitted by an uncompromised protocol participant. Inspecting Fig. 3, we see that this must be a server run.

We know that the server run found in Step 1 freshly generated the value K', and we later determined that $K' = K$. By the freshness assumption, the run in

which $\{|A \hat{} K|\}_{\mathsf{ltk}(B)}$ was transmitted must have been the same as the run in which $\{|B \hat{} K \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(A)}$ was transmitted to A. (See Fig. 6.)

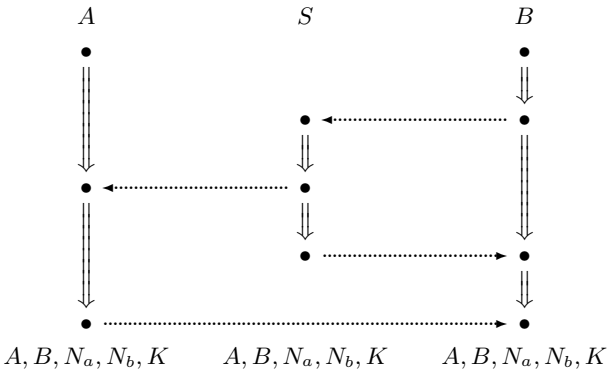

Fig. 6. The resulting shape

Summary. Thus, any run in which the responder B has the behavior shown in on the right in Figure 3 also has one server execution and an initiator execution. All three participants agree on all of the parameters. The session key K cannot have been compromised. We used three assumptions: (1) $\mathsf{ltk}(A), \mathsf{ltk}(B)$ were uncompromised; (2) N_b was freshly chosen; and (3) K will be freshly chosen in every server run.

This argument, which repeatedly used the authentication test idea. corrects the unsound proof given in [8]. The remainder of this paper provides a rigorous theory for this type of shape analysis, and also shows how to mechanize the reasoning. We return to apply our theory to the Yahalom protocol in Section 7.

3 Terms, Strands, and Bundles

Terms (or messages) form a free algebra A, built from atomic terms via constructors. The atoms are partitioned into the types *principals*, *texts*, *keys*, and *nonces*. An inverse operator is defined on keys. There may be additional functions on atoms, such as an injective *public key of* function mapping principals to keys, or an injective *long term shared key of* function mapping pairs of principals to keys. These functions are not constructors, and their results are atoms. For definiteness, we include here functions $\mathsf{pubk}(a), \mathsf{ltk}(a)$ mapping principals to (respectively) their public keys and to a symmetric key shared on a long-term basis with a fixed server S. $\mathsf{pubk}(a)^{-1}$ is a's private key, where $\mathsf{pubk}(a)^{-1} \neq \mathsf{pubk}(a)$. We often write the public key pair as K_a, K_a^{-1}. By contrast, $\mathsf{ltk}(a)^{-1} = \mathsf{ltk}(a)$.

Atoms, written in italics (e.g. a, N_a, K^{-1}), serve as indeterminates (variables). We assume A contains infinitely many atoms of each type. Terms in A are freely built from atoms using *tagged concatenation* and *encryption*. The tags are chosen from a set of constants written in sans serif font (e.g. tag). The tagged concatenation using tag of t_0 and t_1 is written $\mathsf{tag} \hat{} t_0 \hat{} t_1$. Tagged concatenation using the distinguished tag null of t_0 and t_1 is written $t_0 \hat{} t_1$. Encryption takes a term t and an atomic key K, and yields a term as result written $\{|t|\}_K$.

Replacements have only atoms in their range:

Definition 1 (Replacement, Application). A *replacement* is a function α mapping atoms to atoms, such that (1) for every atom a, $\alpha(a)$ is an atom of the same type as a, and (2) α is a homomorphism with respect to the operations on atoms, i.e., $\alpha(K^{-1}) = (\alpha(K))^{-1}$ and $\alpha(\mathsf{pubk}(a)) = \mathsf{pubk}(\alpha(a))$.

The *application* of α to t, written $t \cdot \alpha$, homomorphically extends α's action on atoms. More explicitly, if $t = a$ is an atom, then $a \cdot \alpha = \alpha(a)$; and:

$$(\mathsf{tag} \hat{} t_0 \hat{} t_1) \cdot \alpha = \mathsf{tag} \hat{} (t_0 \cdot \alpha) \hat{} (t_1 \cdot \alpha)$$
$$(\{|t|\}_K) \cdot \alpha = \{|t \cdot \alpha|\}_{K \cdot \alpha}$$

Application distributes through larger objects such as pairing and sets. Thus, $(x, y) \cdot \alpha = (x \cdot \alpha, y \cdot \alpha)$, and $S \cdot \alpha = \{x \cdot \alpha \colon x \in S\}$. If $x \notin \mathsf{A}$ is a simple value such as an integer or a symbol, then $x \cdot \alpha = x$.

Since replacements map atoms to atoms, not to compound terms, unification is very simple. Two terms are unifiable if and only if they have the same abstract syntax tree structure, with the same tags associated with corresponding concatenations, and the same type for atoms at corresponding leaves. To unify t_1, t_2 means to partition the atoms at the leaves; a most general unifier is a finest partition that maps a, b to the same c whenever a appears at the end of a path in t_1 and b appears at the end of the same path in t_2. If two terms t_1, t_2 are unifiable, then $t_1 \cdot \alpha$ and $t_2 \cdot \beta$ are still unifiable.

The direction $+$ means transmission, and the direction $-$ means reception:

Definition 2 (Strand Spaces). A *direction* is one of the symbols $+, -$. A *directed term* is a pair (d, t) with $t \in \mathsf{A}$ and d a direction, normally written $+t, -t$. $(\pm\mathsf{A})^*$ is the set of finite sequences of directed terms.

A *strand space* over A is a structure containing a set Σ and two mappings: a trace mapping $\mathsf{tr} : \Sigma \to (\pm\mathsf{A})^*$ and a replacement application operator $(s, \alpha) \mapsto s \cdot \alpha$ such that (1) $\mathsf{tr}(s \cdot \alpha) = (\mathsf{tr}(s)) \cdot \alpha$, and (2) $s \cdot \alpha = s' \cdot \alpha$ implies $s = s'$.

By condition (2), Σ has infinitely many copies of each strand s, i.e. strands s' with $\mathsf{tr}(s') = \mathsf{tr}(s)$.

Definition 3. A *penetrator strand* has trace of one of the following forms:

M_t: $\langle +t \rangle$ where $t \in$text, principal,nonce
K_K: $\langle +K \rangle$
$\mathsf{C}_{g,h}$: $\langle -g,\ -h,\ +g \hat{} h \rangle$
$\mathsf{S}_{g,h}$: $\langle -g \hat{} h,\ +g,\ +h \rangle$
$\mathsf{E}_{h,K}$: $\langle -K,\ -h,\ +\{|h|\}_K \rangle$
$\mathsf{D}_{h,K}$: $\langle -K^{-1},\ -\{|h|\}_K,\ +h \rangle$.

If s is a penetrator strand, then $s \cdot \alpha$ is a penetrator strand of the same kind.

The *subterm* relation, written $\sqsubset$, is the least reflexive, transitive relation such that (1) $t_0 \sqsubset \mathsf{tag} \,\hat{}\, t_0 \,\hat{}\, t_1$; (2) $t_1 \sqsubset \mathsf{tag} \,\hat{}\, t_0 \,\hat{}\, t_1$; and (3) $t \sqsubset \{|t|\}_K$. Notice, however, $K \not\sqsubset \{|t|\}_K$ unless (anomalously) $K \sqsubset t$. We say that a key K is *used for encryption* in a term t if for some t_0, $\{|t_0|\}_K \sqsubset t$.

A *node* is a pair $n = (s, i)$ where $i \leq \mathsf{length}(\mathsf{tr}(s))$; $\mathsf{strand}(s, i) = s$; and the *direction* and *term* of n are those of $\mathsf{tr}(s)(i)$. We prefer to write $s \downarrow i$ for the node $n = (s, i)$.

A term t *originates* at node n if n is positive, $t \sqsubset \mathrm{term}(n)$, and $t \not\sqsubset \mathrm{term}(m)$ whenever $m \Rightarrow^+ n$. Thus, t originates on n if t is part of a message transmitted on n, and t was neither sent nor received previously on this strand. If a originates on strand s, we write $\mathcal{O}(s, a)$ to refer to the node on which it originates.

A *listener role* is a regular strand $\mathsf{Lsn}[a]$ with trace $\langle -a \rangle$. It documents that a is available on its own to the adversary, unprotected by encryption. Since replacements respect type, atoms of different type must be overheard by different roles. We assume each protocol Π has listener roles $\mathsf{Lsn}[N]$ and $\mathsf{Lsn}[K]$ for nonces and keys respectively, with traces $\langle -N \rangle$ and $\langle -K \rangle$.

Definition 4 (Protocols). A *candidate* $\langle \Pi, n, u \rangle$ consists of: (1) a finite set Π of strands—containing the listener strands $\mathsf{Lsn}[N], \mathsf{Lsn}[K]$—called the *roles* of the protocol; (2) a function n mapping each role r to a finite set of keys n_r, called the non-originating keys of r; and (3) a function u mapping each role r to a finite set of atoms u_r called the uniquely originating atoms of r.

A candidate $\langle \Pi, n, u \rangle$ is a *protocol* if (1) $K \in n_r$ implies that K or K^{-1} is used for encryption on some term of $\mathsf{tr}(r)$, but does not occur in any node of r; and (2) $a \in u_r$ implies that a originates on r, i.e. $\mathcal{O}(r, a)$ is well defined.

The *regular strands* of $\langle \Pi, n, u \rangle$ form the set $\Sigma_\Pi =$

$$\{r \cdot \alpha \colon r \in \Pi \text{ and } \forall a \in u_r, \mathcal{O}(r \cdot \alpha, a \cdot \alpha) = (\mathcal{O}(r, a)) \cdot \alpha\}.$$

Example 5 (Yahalom Protocol). The Yahalom protocol has a set Π_Y of roles containing the three roles shown in Fig. 3 and two listener roles, to hear nonces and keys. For each $r \in \Pi_Y$, $n_r = \emptyset$. For the server role $\mathsf{Serv} \in \Pi_Y$, $u_{\mathsf{Serv}} = \{K\}$, and for the other roles $r \in \Pi_Y$, $u_r = n_r = \emptyset$.

The set $\mathcal{N}$ of all nodes forms a directed graph $\mathcal{G} = \langle \mathcal{N}, (\rightarrow \cup \Rightarrow) \rangle$ with edges $n_1 \rightarrow n_2$ for communication (with the same term, directed from positive to negative node) and $n_1 \Rightarrow n_2$ for succession on the same strand.

Definition 6 (Bundle). A finite acyclic subgraph $\mathcal{B} = \langle \mathcal{N}_\mathcal{B}, (\rightarrow_\mathcal{B} \cup \Rightarrow_\mathcal{B}) \rangle$ of $\mathcal{G}$ is a *bundle* if (1) if $n_2 \in \mathcal{N}_\mathcal{B}$ is negative, then there is a unique $n_1 \in \mathcal{N}_\mathcal{B}$ with $n_1 \rightarrow_\mathcal{B} n_2$; and (2) if $n_2 \in \mathcal{N}_\mathcal{B}$ and $n_1 \Rightarrow n_2$, then $n_1 \Rightarrow_\mathcal{B} n_2$. When $\mathcal{B}$ is a bundle, $\preceq_\mathcal{B}$ is the reflexive, transitive closure of $(\rightarrow_\mathcal{B} \cup \Rightarrow_\mathcal{B})$.

A bundle $\mathcal{B}$ is *over* $\langle \Pi, n, u \rangle$ if for every $s \downarrow i \in \mathcal{B}$, (1) either $s \in \Sigma_\Pi$ or s is a penetrator strand; (2) if $s = r \cdot \alpha$ and $a \in n_r \cdot \alpha$, then a does not originate in $\mathcal{B}$; and (3) if $s = r \cdot \alpha$ and $a \in u_r \cdot \alpha$, then a originates at most once in $\mathcal{B}$.

Example 7. Fig. 1 is a bundle if we connect arrows with matching labels. Fig. 2 is a bundle if we replace C with B and then connect arrows with matching labels. Alternatively, it becomes a bundle by adding penetrator strands to unpack the values encrypted with K_C and repackage the values, encrypting with K_B.

We say that a strand s is *in* $\mathcal{B}$ if s has at least one node in $\mathcal{B}$.

Proposition 1. *Let $\mathcal{B}$ be a bundle. $\preceq_{\mathcal{B}}$ is a well-founded partial order. Every non-empty set of nodes of $\mathcal{B}$ has $\preceq_{\mathcal{B}}$-minimal members.*

Let α be a replacement. Suppose for every regular strand $s = r \cdot \beta$ in $\mathcal{B}$, for every $b \in u_r \cdot \beta$, we have $(\mathcal{O}(s, b)) \cdot \alpha = \mathcal{O}(s \cdot \alpha, b \cdot \alpha)$. Then $\mathcal{B} \cdot \alpha$ is a bundle.

4 Preskeletons, Skeletons, and Homomorphisms

A preskeleton is potentially the regular (non-penetrator) part of a bundle or of some portion of a bundle. It is annotated with additional information, indicating order relations among nodes, uniquely originating atoms, and non-originating atoms. We say that an atom a *occurs* in a set nodes of nodes if for some $n \in \mathsf{nodes}$, $a \sqsubset \mathrm{term}(n)$. A key K is *used* in nodes if for some $n \in \mathsf{nodes}$, $\{\!|t|\!\}_K \sqsubset \mathrm{term}(n)$. We say that a key K is *mentioned in* nodes if K or K^{-1} either occurs or is used in nodes. For a non-key a, a is mentioned if it occurs.

Definition 8. A four-tuple $\mathbb{A} = (\mathsf{nodes}, \preceq, \mathsf{non}, \mathsf{unique})$ is a *preskeleton* if:

1. nodes is a finite set of regular nodes; $n_1 \in \mathsf{nodes}$ and $n_0 \Rightarrow^+ n_1$ implies $n_0 \in \mathsf{nodes}$;
2. $\preceq$ is a partial ordering on nodes such that $n_0 \Rightarrow^+ n_1$ implies $n_0 \preceq n_1$;
3. non is a set of keys, and for all $K \in \mathsf{non}$, either K or K^{-1} is used in nodes;

3′. for all $K \in \mathsf{non}$, K does not occur in nodes;

4. unique is a set of atoms, and for all $a \in \mathsf{unique}$, a occurs in nodes.

A preskeleton $\mathbb{A}$ is a *skeleton* if in addition:

4′. $a \in \mathsf{unique}$ implies a originates at no more than one node in nodes.

We select components of a preskeleton using subscripts, so, in $\mathbb{A} = (N, R, S, S')$, $\preceq_{\mathbb{A}}$ means R and $\mathsf{unique}_{\mathbb{A}}$ means S'. $\mathbb{A}$ need not contain all of the nodes of a strand, just some initial subsequence. We write $n \in \mathbb{A}$ to mean $n \in \mathsf{nodes}_{\mathbb{A}}$, and we say that a strand s is in $\mathbb{A}$ when at least one node of s is in $\mathbb{A}$. The $\mathbb{A}$-height of s is the largest i with $s \downarrow i \in \mathbb{A}$. By Clauses 3, 4, $\mathsf{unique}_{\mathbb{A}} \cap \mathsf{non}_{\mathbb{A}} = \emptyset$.

4.1 Skeletons and Bundles

Bundles correspond to certain skeletons:

Definition 9. Bundle $\mathcal{B}$ *realizes* skeleton $\mathbb{A}$ if:

1. The nodes of $\mathbb{A}$ are the regular nodes $n \in \mathcal{B}$.

2. $n \preceq_{\mathbb{A}} n'$ just in case $n, n' \in \mathsf{nodes}_{\mathbb{A}}$ and $n \preceq_{\mathcal{B}} n'$.
3. $K \in \mathsf{non}_{\mathbb{A}}$ iff case K or K^{-1} is used in $\mathsf{nodes}_{\mathbb{A}}$ but K occurs nowhere in $\mathcal{B}$.
4. $a \in \mathsf{unique}_{\mathbb{A}}$ iff a originates uniquely in $\mathcal{B}$.

The *skeleton* of $\mathcal{B}$ is the skeleton that it realizes. The skeleton of $\mathcal{B}$, written $\mathsf{skeleton}(\mathcal{B})$, is uniquely determined. $\mathbb{A}$ is *realized* if some $\mathcal{B}$ realizes it.

Two bundles $\mathcal{B}, \mathcal{B}'$ are *similar*, written $\mathcal{B} \sim_{\mathsf{L}} \mathcal{B}'$, if they differ only in what listener strands they contain. Two realized skeletons $\mathbb{A}, \mathbb{A}'$ are *similar*, written $\mathbb{A} \sim_{\mathsf{L}} \mathbb{A}'$, if for some $\mathcal{B}, \mathcal{B}'$ with $\mathcal{B} \sim_{\mathsf{L}} \mathcal{B}'$, $\mathbb{A} = \mathsf{skeleton}(\mathcal{B})$ and $\mathbb{A}' = \mathsf{skeleton}(\mathcal{B}')$.

By condition (4), $\mathcal{B}$ does not realize $\mathbb{A}$ if $\mathbb{A}$ is a preskeleton but not a skeleton. Given a skeleton $\mathbb{A}$, methods derived from [9] determine whether $\mathbb{A}$ is realized.

4.2 Homomorphisms

When $\mathbb{A}$ is a preskeleton, we may apply a substitution α to it, subject to the same condition as in Prop. 1. Namely, suppose α is a replacement, and suppose that for each regular strand $s = r \cdot \beta$ such that s has nodes in $\mathbb{A}$, and for each atom $b \in u_r \cdot \beta$,

$$(\mathcal{O}(s, b)) \cdot \alpha = \mathcal{O}(s \cdot \alpha, b \cdot \alpha).$$

Then $\mathbb{A} \cdot \alpha$ is a well defined object. However, it is not a preskeleton when $x \cdot \alpha = y \cdot \alpha$ where $x \in \mathsf{non}_{\mathbb{A}}$ while y occurs in $\mathbb{A}$. In this case, no further identifications can restore the preskeleton property. So we are interested only in replacements with the property that $x \cdot \alpha = y \cdot \alpha$ and $x \in \mathsf{non}_{\mathbb{A}}$ implies y does not occur in $\mathbb{A}$. On this condition, $\mathbb{A} \cdot \alpha$ is a preskeleton.

However, $\mathbb{A}$ may be a skeleton, while $\mathbb{A} \cdot \alpha$ is a preskeleton but not a skeleton. For instance, this occurs when two distinct atoms $a_1, a_2 \in \mathsf{unique}_{\mathbb{A}}$, but $a_1 \cdot \alpha = a_2 \cdot \alpha$. There will then be two nodes n_1, n_2 that are points of origination for a_1, a_2 respectively. Hence, the two nodes $n_i \cdot \alpha$ are both points of origination for the common value $a_i \cdot \alpha$. We may be able, though, to restore the skeleton unique origination property (4′) by a mapping ϕ that carries these two nodes to a common node. This will be possible only if the terms on them are the same, and likewise for the other nodes in $\mathbb{A}$ on the same strands. We regard ϕ, α as an information-preserving, or more specifically information-increasing, map. It has added the information that a_1, a_2, which could have been distinct, are in fact the same, and thus the nodes n_1, n_2, which could have been distinct, must also be identified.

Definition 10. Let $\mathbb{A}_0, \mathbb{A}_1$ be preskeletons, α a replacement, $\phi\colon \mathsf{nodes}_{\mathbb{A}_0} \to \mathsf{nodes}_{\mathbb{A}_1}$. $H = [\phi, \alpha]$ is a *homomorphism* if

1a. For all $n \in \mathbb{A}_0$, $\mathrm{term}(\phi(n)) = \mathrm{term}(n) \cdot \alpha$;
1b. For all s, i, if $s \downarrow i \in \mathbb{A}$ then there is an s' s.t. for all $j \leq i$, $\phi(s \downarrow j) = (s', j)$;
2. $n \preceq_{\mathbb{A}_0} m$ implies $\phi(n) \preceq_{\mathbb{A}_1} \phi(m)$;
3. $\mathsf{non}_{\mathbb{A}_0} \cdot \alpha \subset \mathsf{non}_{\mathbb{A}_1}$;
4. $\mathsf{unique}_{\mathbb{A}_0} \cdot \alpha \subset \mathsf{unique}_{\mathbb{A}_1}$; and $\mathcal{O}(s, a) = \mathcal{O}(s', a \cdot \alpha)$ whenever $a \in \mathsf{unique}_{\mathbb{A}_0}$, $\mathcal{O}(s, a) \in \mathbb{A}_0$, and $\phi(s \downarrow j) = s' \downarrow j$.

We write $H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ when H is a homomorphism from $\mathbb{A}$ to $\mathbb{A}'$. When $a \cdot \alpha = a \cdot \alpha'$ for every a that occurs or is used for encryption in $\mathsf{dom}(\phi)$, then $[\phi, \alpha] = [\phi, \alpha']$; i.e., $[\phi, \alpha]$ is the equivalence class of pairs under this relation.

The condition on $\mathcal{O}$ in Clause 4 avoids a kind of degeneracy, in which a point of origination is destroyed for some atom $a \in \mathsf{unique}_{\mathbb{A}_0}$ by identifying a with a value occurring earlier on the strand. We stipulate that such a map is not a homomorphism. The condition for $[\phi, \alpha] = [\phi, \alpha']$ implies that the action of α on atoms not mentioned in the $\mathbb{A}_0$ is irrelevant.

When transforming a preskeleton $\mathbb{A}$ into a skeleton, one identifies nodes n, n' if some $a \in \mathsf{unique}_{\mathbb{A}}$ originates on both; to do so, one may need to unify additional atoms that appear in both $\mathrm{term}(n), \mathrm{term}(n')$. Although this process may cascade, when success is possible, there is a canonical (universal) way to succeed, as follows from the results in Appendix A:

Proposition 2. *Suppose $\mathbb{A}$ is a preskeleton and $\mathbb{A}'$ is a skeleton where $H\colon \mathbb{A} \mapsto \mathbb{A}'$. There exists a homomorphism $G_{\mathbb{A}}$ and a skeleton $\mathbb{A}_0$ such that $G_{\mathbb{A}}\colon \mathbb{A} \mapsto \mathbb{A}_0$ and, for every skeleton $\mathbb{A}_1$ and homomorphism $H_1\colon \mathbb{A} \mapsto \mathbb{A}_1$, for some H, $H_1 = H \circ G_{\mathbb{A}}$. $G_{\mathbb{A}}$ and $\mathbb{A}_0$ are unique to within isomorphism.*

We call this universal map $G_{\mathbb{A}}$ (or sometimes its target $\mathbb{A}_0$) the *hull* of $\mathbb{A}$, $\mathsf{hull}(\mathbb{A})$.

4.3 Shapes

Definition 11 (Shape). $\mathbb{A}_0$ is *nodewise less than or equal to* $\mathbb{A}_1$ if for some homomorphism $[\phi, \alpha]\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$, ϕ is an injective function on the nodes of $\mathbb{A}_0$. We say $\mathbb{A}$ is *nodewise minimal* in a set S of skeletons if $\mathbb{A}$ is nodewise less than or equal to all $\mathbb{A}_1 \in S$.

$\mathbb{A}'$ *interpolates into* $H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ if $H = H'' \circ H'$ where $H'\colon \mathbb{A}_0 \mapsto \mathbb{A}'$.

$H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ is a *shape for* $\mathbb{A}_0$ if $\mathbb{A}_1$ is nodewise minimal among realized skeletons that interpolate into H.

$\mathbb{A}_0, \mathbb{A}_1$ are isomorphic if each is nodewise less than or equal to the other. If we speak of a skeleton $\mathbb{A}$ as a shape, we mean that it is the target $\mathbb{A}_1$ of some shape $H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$, where a particular $\mathbb{A}_0$ is understood from the context.

Proposition 3. *Let $H = [\phi, \alpha]\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$. The set $\mathcal{A}$ of realized skeletons nodewise less than or equal to $\mathbb{A}_1$ that interpolate into H is finite (up to isomorphism). If $\mathbb{A}_1$ is realized, then at least one of the interpolants H' is a shape for $\mathbb{A}_0$.*

Proof. We generate $\mathcal{A}$ by choosing, for each node $n \in (\mathbb{A}_1 \setminus \phi(\mathbb{A}_0))$, whether to omit it and all nodes later than n on the same strand.

If a is mentioned multiple times in $\mathbb{A}_1$, we partition the locations at which it is mentioned. We will leave unchanged one equivalence class of locations, and we will replace each other class with a new atom, unmentioned in $\mathbb{A}_1$. A partition is compatible with H when, if $b \cdot \alpha = a$ and $m = \phi(n)$, then for any pair of locations of b in $\mathrm{term}(m)$, the corresponding locations of a in $\mathrm{term}(n)$ are in the same equivalence class.

The set $\mathcal{A}$ of interpolants into H (to within isomorphism) are the skeletons we get given a choice of nodes to omit and a compatible partition. $\mathbb{A}_1 \in \mathcal{A}$, so if $\mathbb{A}_1$ is realized, $\mathcal{A}$ has realized members, hence also minimal realized members.

If $\mathbb{A}_1$ is realized and contains listener strands, and $\mathbb{A}$ results when we omit some of the listener strands, then $\mathbb{A}$ is realized and $\mathbb{A} \sim_{\mathsf{L}} \mathbb{A}_1$. In particular, $\mathbb{A}$ is nodewise less than or equal to $\mathbb{A}_1$. A minimal member of $\mathcal{A}$ will omit all of the listener strands, which is why they do not appear in Figs. 1–2.

Given a skeleton $\mathbb{A}_0$ as "starting point," we would like to find all the homomorphisms $H \colon \mathbb{A}_0 \mapsto \mathbb{A}$ that lead from $\mathbb{A}_0$ to a shape $\mathbb{A}$. If we find homomorphisms from $\mathbb{A}_0$ to realized skeletons $\mathbb{A}_1$, then Prop. 3 tells us how to obtain one or more shapes from each of these realized skeletons. We are thus most interested in homomorphisms H that do not unnecessarily identify occurrences of atoms, as we will try to distinguish the different uses of the same atom in $\mathbb{A}_1$ to find nodewise minimal members of $\mathcal{A}$.

Our search is finished when more realized skeletons cannot yield any shapes we have not yet encountered.

5 The Authentication Tests

To direct the process of searching for realized skeletons, we use the *authentication tests* [9] in a strengthened and simplified form.

We say that t_0 *occurs only within* S in t, where S is a set of terms, if:

1. $t_0 \not\sqsubset t$; or
2. $t \in S$; or
3. $t \neq t_0$ and either (3a) $t = \{\!|t_1|\!\}_K$ and t_0 occurs only within S in t_1; or (3b) $t = \mathsf{tag} \mathbin{\hat{}} t_1 \mathbin{\hat{}} t_2$ and t_0 occurs only within S in each t_i $(i = 1, 2)$.

So t_0 occurs only within S in t if in the abstract syntax tree, every path from the root t to an occurrence of t_0 as a subterm of t traverses some $t_1 \in S$ before reaching t_0. On the other hand, t_0 *occurs outside* S *in* t if t_0 does not occur only within S in t. This means that $t_0 \sqsubset t$ and there is a path from the root to an occurrence of t_0 as a subterm of t that traverses no $t_1 \in S$.

5.1 The Tests in Bundles

We say that a is *protected* in $\mathcal{B}$ iff $\mathrm{term}(n) \neq a$ for all $n \in \mathcal{B}$. Equivalently, a is protected in $\mathcal{B}$ iff the listener strand for a is not in $\mathcal{B}'$ for any $\mathcal{B}' \sim_{\mathsf{L}} \mathcal{B}$; that is, $(\mathsf{Lsn}[a] \downarrow 1) \notin \mathcal{B}'$.

We say that a is *protected up to* m in $\mathcal{B}$ iff, for all $n \in \mathcal{B}$, if $\mathrm{term}(n) = a$ then $m \prec_{\mathcal{B}} n$. We write $a \in \mathsf{Prot}_m(\mathcal{B})$ if a is protected up to m in $\mathcal{B}$.

By the definitions of the penetrator strands for encryption and decryption (Definition 3), if the adversary uses K for encryption or decryption anywhere in $\mathcal{B}$, then K is not protected in $\mathcal{B}$. Thus, the adversary cannot create any encrypted term with a protected key K. If K^{-1} is protected, it cannot decrypt any term encrypted with K. If a key is protected up to a negative node m, then the adversary cannot use that key to prepare the term received on m.

Proposition 4 (Outgoing Authentication Test). *Suppose that* $n_0, n_1 \in \mathcal{B}$, *and*

$$S \subset \{\{|t|\}_K : K^{-1} \in \mathsf{Prot}_{n_1}(\mathcal{B})\}.$$

Suppose that a *originates uniquely in* $\mathcal{B}$ *at node* n_0 *and occurs only within* S *in* $term(n_0)$, *but* a *occurs outside* S *in* $term(n_1)$.

There is an integer i *and a regular strand* $s \in \Sigma_\Pi$ *such that* $m_1 = s \downarrow i \in \mathcal{B}$ *is positive, and* i *is the least integer* k *such that* a *occurs outside* S *in* $term(s \downarrow k)$. *Moreover, there is a node* $m_0 = s \downarrow j$ *with* $j < i$ *such that* $a \sqsubset term(s \downarrow j)$, *and* $n_0 \preceq_{\mathcal{B}} m_0 \Rightarrow^+ m_1 \preceq_{\mathcal{B}} n_1$.

Proof. Apply Prop. 1 to

$$T = \{m : m \preceq_{\mathcal{B}} n_1 \text{ and } a \text{ occurs outside } S \text{ in } \mathrm{term}(m)\}.$$

$n_1 \in T$, so T has $\preceq_{\mathcal{B}}$-minimal members m_1. Since keys K used in S have $K^{-1} \in \mathsf{Prot}(\mathcal{B})$, m_1 cannot lie on a decryption penetrator D-strand. By the assumptions, a does not originate on m_1, so that m_1 does not lie on a M-strand or K-strand. By the definitions of S and "occurs only within," m_1 does not lie on a S-, C-, or E-strand. Thus, m_1 lies on some $s \in \Sigma_\Pi$ at some index i.

In the Outgoing Authentication Test, we call $m_0 \Rightarrow^+ m_1$ an *outgoing transforming edge* for a, S. It transforms the occurrence of a from lying only within S to occurring outside it. We call (n_0, n_1) an *outgoing test pair for* a, S when these nodes satisfy the condition in the first paragraph of the proposition. When we do not know the set $\mathsf{Prot}_{n_1}(\mathcal{B})$, but consider a candidate set of atoms X, we speak of an *outgoing test pair for* a, S, X.

Proposition 5 (Incoming Authentication Test). *Suppose that* $n_1 \in \mathcal{B}$ *is negative,* $t = \{|t_0|\}_K \sqsubset term(n_1)$, *and* $K \in \mathsf{Prot}(\mathcal{B})$. *There exists a regular* $m_1 \prec n_1$ *such that* t *originates at* m_1.

Proof. Apply Prop. 1 to $T = \{m : m \preceq_{\mathcal{B}} n_1 \text{ and } t \sqsubset \mathrm{term}(m)\}$. A minimal member $m_1 \in T$ does not lie on a penetrator E-strand because $K \in \mathsf{Prot}(\mathcal{B})$.

Here we refer to m_1 as an *incoming transforming node*, and in the solicited case we call $m_0 \Rightarrow^+ m_1$ an *incoming transforming edge*. We call n_1 an incoming test node.

5.2 The Tests in Skeletons

Since these theorems hold for all bundles, and concern only the regular behavior within the bundles, they hold for all realized skeletons. Thus, roughly speaking, any homomorphism $H : \mathbb{A}_0 \mapsto \mathbb{A}_1$ where $\mathbb{A}_1$ is realized must add a transforming edge when $\mathbb{A}_0$ does not already contain one. Indeed, we can regard H as a composition $H'' \circ H'$ where H' adds the transforming edge right away, and H'' does whatever else is needed to construct $\mathbb{A}_1$.

Definition 12 (Augmentations, Contractions). 1. The inclusion $[\mathsf{id}, \mathsf{id}] \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ is an *augmentation* if:
   (a) $\mathsf{nodes}_{\mathbb{A}_1} \setminus \mathsf{nodes}_{\mathbb{A}_0} = \{s \downarrow j \colon j \leq i\}$ for some $s = r \cdot \alpha$;
   (b) $\preceq_{\mathbb{A}_1}$ is the transitive closure of (a) $\preceq_{\mathbb{A}_0}$; (b) the strand ordering of s up to i; and (c) pairs (n, m) or (n, m) with $n \in \mathsf{nodes}_{\mathbb{A}_1}$, $m = s \downarrow j$, and $j \leq i$.
   (c) $\mathsf{non}_{\mathbb{A}_1} = \mathsf{non}_{\mathbb{A}_1} \cup (n_r \cdot \alpha)$; and
   (d) $\mathsf{unique}_{\mathbb{A}_1} = \mathsf{unique}_{\mathbb{A}_1} \cup (u_r \cdot \alpha)$.
2. An augmentation $H \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ is an *outgoing augmentation* if there exists an outgoing test edge $n_0, n_1 \in \mathbb{A}_0$ with no outgoing transforming edge in $\mathbb{A}_0$, and $s \downarrow 1 \Rightarrow^* m_0 \Rightarrow^+ s \downarrow i$, where $m_0 \Rightarrow^+ s \downarrow i$ is the earliest transforming edge for this test on s. The additional pairs in the ordering (clause 2c) are the pairs (n_0, m_0) and $((s \downarrow i), n_1)$.
3. It is an *incoming augmentation* if it adds an incoming transforming edge for an incoming test node in $\mathbb{A}_0$. The pair (m_1, n_1) in the notation of Prop. 5 is the additional pair in the ordering.
4. It is a *listener augmentation for a* if it adds a listener strand $\mathsf{Lsn}[a]$, with no pairs added to the ordering. Thus, $\mathbb{A}_1 \sim_{\mathsf{L}} \mathbb{A}_0$ if $\mathbb{A}_1$ results by a listener augmentation.
5. A replacement α is a *contraction* for $\mathbb{A}$ if there are two distinct atoms a, b mentioned in $\mathbb{A}$ such that $a \cdot \alpha = b \cdot \alpha$. We write $\mathsf{hull}_\alpha(\mathbb{A})$ for the canonical homomorphism from $\mathbb{A}$ to $\mathsf{hull}(\mathbb{A} \cdot \alpha)$. (See Prop. 2.)

We can now state the search-oriented version of Prop. 4. It states that when a skeleton $\mathbb{A}_0$ with an unsolved outgoing transformed pair can lead to a realized skeleton $\mathbb{A}_1$, we can get there by starting out with one of three kinds of steps: (1) an outgoing augmentation, (2) a contraction, or (3) adding a listener strand to witness for the fact that one of the relevant keys is in fact *not* properly protected by the time we reach $\mathbb{A}_1$.

Theorem 1 (Outgoing Augmentation). *Let $H \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$, where $\mathbb{A}_1$ is realized. Let X be a set of keys, and let $n_0, n_1 \in \mathbb{A}_0$ be an outgoing test pair for a, S, X, for which $\mathbb{A}_0$ contains no transforming edge. At least one of the following holds:*

1. *$H = H'' \circ \mathsf{hull}_\alpha(\mathbb{A}_0)$ for some contraction α;*
2. *$H = H'' \circ H'$, where H' is some outgoing augmentation for a, S, X;*
3. *There is a listener augmentation $H' \colon \mathbb{A}_0 \mapsto \mathbb{A}'_0$ for some $K \in X$, and a homomorphism $H'' \colon \mathbb{A}'_0 \mapsto \mathbb{A}'_1$ such that: (a) $\mathbb{A}'_1$ is realized, (b) $\mathbb{A}'_1 \sim_{\mathsf{L}} \mathbb{A}_1$, and (c) $H'' \circ H' = I \circ H$, where I is an inclusion homomorphism.*

Proof. Assuming $H = [\phi, \alpha] \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ with $\mathbb{A}_1$ realized, say with $\mathsf{skeleton}(\mathcal{B}) = \mathbb{A}_1$, we have the following possibilities. If α contracts any atoms, then we may factor H into a contraction followed by some remainder H'' (clause 1).

If α does not contract any atoms, then $(\phi(n_0), \phi(n_1))$ is an outgoing test pair for $a \cdot \alpha, S \cdot \alpha, X \cdot \alpha$. There are now two cases. First, suppose $X \cdot \alpha \subseteq \mathsf{Prot}_{\phi(n_1)}(\mathcal{B})$. Then we may apply Prop. 4 to infer that $\mathcal{B}$ and thus also $\mathbb{A}_1$ contains an outgoing

transforming edge $m_0 \Rightarrow^+ m_1$ for $a \cdot \alpha, S \cdot \alpha$. Since α is injective on atoms mentioned in $\mathbb{A}_0$, we may augment $\mathbb{A}_0$ with $(m_0 \cdot \alpha^{-1}) \Rightarrow^+ (m_1 \cdot \alpha^{-1})$.

Second, if there is some $a \in X$ such that $a \cdot \alpha \notin \mathsf{Prot}_{\phi(n_1)}\mathcal{B}$, then there is $\mathbb{A}'_1 \sim_{\mathsf{L}} \mathbb{A}_1$ such that $\mathbb{A}'_1$ contains $\mathsf{Lsn}[a \cdot \alpha]$, and $\phi(n_1) \not\preceq (\mathsf{Lsn}[a \cdot \alpha]) \downarrow 1$. Hence, clause 3 is satisfied.

In applying Theorem 1, we prefer to apply Clauses 2, 3 if possible; unnecessary contractions must simply be un-contracted using Prop. 3. In particular, we use a contraction α only if either (1) $n_0 \cdot \alpha, n_1 \cdot \alpha$ is no longer an outgoing transformed pair, or else (2) for some candidate outgoing augmentation, $n_0 \cdot \alpha, n_1 \cdot \alpha$ is the most general version of the test that it solves. The latter may occur when the protocol role mentions the same atom at several locations where different atoms are mentioned in n_0, n_1; α must then identify these atoms.

We can now see something missing with the analysis in Section 2.2. Theorem 1 always interpolates the nodes m_0, m_1 between a's point of origination and the node n_1 in which it occurs outside S. Thus, the repeated of it in Steps 1 and 2 place the transforming nodes between B's first transmission and final reception, but do not determine any order for the server strand and the initiator strand. As we shall see in Section 7, we may introduce the strands in reverse order, and establish that the server behavior causally preceded the initiator's second action. However, step 3 works out differently. The analysis of Yahalom in [8] is unsound, because the version of the authentication tests we used at the time did not contain a set of terms S. Effectively, it covered the case in which S is a singleton, which makes Step 2 impossible to state.

Incoming augmentations are similar to outgoing ones, except that the relevant keys are only those used for encryption in the test node:

Theorem 2 (Incoming Augmentation). *Let $H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$, where $\mathbb{A}_1$ is realized. Let $n_1 \in \mathbb{A}_0$ be a negative node and $\{|t_0|\}_K \sqsubset \mathit{term}(n_1)$. If $\{|t_0|\}_K$ originates nowhere in $\mathbb{A}_0$, then either:*

1. *$H = H'' \circ H'$, where H' is an incoming augmentation originating $\{|t_0|\}_K$; or*

2. *There is a listener augmentation $H'\colon \mathbb{A}_0 \mapsto \mathbb{A}'_0$ for K, and a homomorphism $H''\colon \mathbb{A}'_0 \mapsto \mathbb{A}'_1$ such that: (a) $\mathbb{A}'_1$ is realized, (b) $\mathbb{A}'_1 \sim_{\mathsf{L}} \mathbb{A}_1$, and (c) $H'' \circ H' = I \circ H$, where I is an inclusion homomorphism.*

Here we use a contraction α only when α is needed to make an incoming augmentation apply. A contraction never eliminates an incoming test node.

When $a \sqsubset \mathrm{term}(m)$, where $a \in \mathsf{unique}_{\mathbb{A}_0}$ and $m \in \mathbb{A}_0$, and a originates at $n \in \mathbb{A}_0$, then n will precede m in any bundle accessible from $\mathbb{A}_0$. That is, if $H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ where the latter is realized, then H factors through H' which maps $\mathbb{A}_0$ to the order enrichment $\mathbb{A}'_0$, where $\preceq_{\mathbb{A}'_0}$ is the transitive closure of $(\preceq_{\mathbb{A}_0} \cup (n, m))$. We will rely on this implicitly in what follows. When we need to be explicit about this, to say that a skeleton needs no further enrichment of this kind, we will say that its *order reflects origination*.

5.3 Completeness of the Authentication Tests

If a skeleton $\mathbb{A}$ is not realized, does it necessarily contain an outgoing transformed edge or an incoming transformed node? Yes, it does, although to make this precise we must be careful about which atoms are protected, as this is not explicit in an unrealized skeleton.

Definition 13 (Penetrator web). Let $G = \langle \mathcal{N}_G, (\rightarrow_G \cup \Rightarrow_G) \rangle$ be a finite acyclic subgraph of $\langle \mathcal{N}, (\rightarrow \cup \Rightarrow) \rangle$ such that $\mathcal{N}_G$ consists entirely of penetrator nodes. G is a *penetrator web* with support S and result R if S and R are sets of terms and moreover:

1. If $n_2 \in \mathcal{N}_G$ is negative, then either $\mathrm{term}(n_2) \in S$ or there is a unique n_1 such that $n_1 \rightarrow_G n_2$.
2. If $n_2 \in \mathcal{N}_G$ and $n_1 \Rightarrow n_2$ then $n_1 \Rightarrow_G n_2$.
3. For each $t \in R$, either $t \in S$ or for some positive $n \in \mathcal{N}_G$, $\mathrm{term}(n) = t$.

If $n \in \mathcal{B}$ is a negative node, then $\mathcal{B}$ includes a penetrator web G with result $R_G = \{\mathrm{term}(n)\}$. Its support $S_G = \{\mathrm{term}(m) \colon m \text{ is positive regular and } m \prec_{\mathcal{B}} n\}$. We write the set of positive regular nodes preceding a node n as $\mathsf{support}(n)$.

Definition 14. A term t is *penetrator-derivable before* n in $\mathbb{A}$ if there is a penetrator web G with $t \in R_G$ such that:

1. $S_G \subset \mathsf{support}(n)$;
2. If $K \in \mathsf{non}_{\mathbb{A}}$, K does not originate in G_n; and
3. If $a \in \mathsf{unique}_{\mathbb{A}}$ and a originates in $\mathbb{A}$, then a does not originate in G_n.

Proposition 6. *A skeleton $\mathbb{A}$ is realized iff, for every negative $n \in \mathbb{A}$, term(n) is penetrator-derivable before n in $\mathbb{A}$.*

Proposition 7. *Suppose that $\preceq_{\mathbb{A}}$ reflects origination, and there exists some $H \colon \mathbb{A} \mapsto \mathbb{A}'$ where $\mathbb{A}'$ is realized. If term(n) is not penetrator-derivable before n in $\mathbb{A}$, then either:*

1. *n is an incoming transformed node for some $K \in \mathsf{non}_{\mathbb{A}} \cup \mathsf{unique}_{\mathbb{A}}$; or else*
2. *(m, n) is an outgoing transformed pair with respect to a, S, X for (i) some $a \in \mathsf{unique}_{\mathbb{A}}$ originating at a node $m \in \mathbb{A}$; (ii) some set S of encrypted terms such that a occurs only within S in $\mathsf{support}(n)$; and (iii) some set of keys $X \subset \mathsf{non}_{\mathbb{A}} \cup \mathsf{unique}_{\mathbb{A}}$ such that for each $K \in X$, K^{-1} is used for encryption in $\mathsf{support}(n)$.*

Proof. By structural induction on the terms $n, \mathsf{support}(n)$. *Base Case:* Suppose that $\mathrm{term}(m)$ is an atom, whenever $m \in \mathsf{support}(n)$ or $m = n$.

Theorem 3 (Authentication Tests Completeness). *Let $H \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ be a shape. $H = H_k \circ H_{k-1} \circ \ldots \circ H_1 \circ H_0$ for some sequence of homomorphisms $\{H_i\}_{0 \leq i \leq k}$, where*

1. *H_0 is a surjective homomorphism from $\mathbb{A}_0$ onto a substructure (possibly the identity); and*
2. *For each i with $1 \leq i \leq k$, H_i is a contraction or an augmentation as in Theorem 1, Clause 1 or Clause 2, or else Theorem 2, Clause 1.*

Proof. By induction on the length k of the decomposition.

5.4 A Pruning Condition

Some augmentations make progress toward realized skeletons, and other augmentations make no progress, because although they introduce a strand, that new strand is a redundant copy of an existing strand. We can prune away these augmentations, and ignore them when searching for shapes.

We say $\mathbb{A}'_0$ *augments* $\mathbb{A}_0$ *with a copy of* s if $\mathbb{A}'_0$ results from $\mathbb{A}_0$ by an augmentation with a strand s' such that: (1) $\mathsf{nodes}_{\mathbb{A}'_0} \setminus \mathsf{nodes}_{\mathbb{A}_0} = \{s' \downarrow j : j \leq i\}$ for some i; (2) there is an idempotent $I_0 = [\psi_0, \beta_0] \colon \mathbb{A}'_0 \mapsto \mathbb{A}_0$ with $\psi_0(s' \downarrow j) = s \downarrow j$.

Proposition 8. *Suppose $\mathbb{A}'_0$ augments $\mathbb{A}_0$ with a copy of s, namely s'. Let $H' = [\phi, \alpha] \colon \mathbb{A}'_0 \mapsto \mathbb{A}'_1$ with $\mathbb{A}'_1$ realized, where $\phi(s' \downarrow j) \neq \phi(s \downarrow j)$. Then there exists $H \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ such that $\mathbb{A}_1$ is realized and properly nodewise less than $\mathbb{A}'_1$.*

Hence, letting $H'' = [\phi, \alpha] \colon \mathbb{A}'_0 \mapsto \mathbb{A}'_1$ be a shape, $\phi(s' \downarrow j) = \phi(s \downarrow j)$.

Proof. Let G map $\mathbb{A}'_0$ to a shape interposed into H'. By Corollary 3, we may write G as a sequence of contractions and augmentations, after taking a substructure that identifies (at least) s with s'. Etc.

6 A Search Strategy

In mechanically searching for shapes from a given starting point, we are essentially building up a number of relations, of which the most important is step, which represents a single application of one of the authentication tests.

step The relation $\mathsf{step}(\mathbb{A}_0, \mathbb{A}_1)$ holds if $\mathbb{A}_1$ results by a contraction, an outgoing augmentation, or a listener augmentation, as in Theorem 1, Clauses 1, 2, and 3 respectively; or by an incoming augmentation or a listener augmentation as in Theorem 2, clauses 1 and 2 respectively. We write step^* for the reflexive, transitive closure of step.

$\mathsf{realized}$ We write $\mathsf{realized}(\mathbb{A})$ to mean that $\mathbb{A}$ is realized, which we can determine directly.

shape We write $\mathsf{shape}(\mathbb{A}, \mathbb{A}')$ to mean that $\mathbb{A}'$ is a shape of the realized skeleton $\mathbb{A}$. By this we mean that $\mathbb{A}'$ is node minimal among realized skeletons with a node-injective $H \colon \mathbb{A}' \mapsto \mathbb{A}$. So $\mathsf{shape}(\mathbb{A}, \mathbb{A}')$ implies $\mathsf{realized}(\mathbb{A})$ and $\mathsf{realized}(\mathbb{A}')$. We determine shapes as described in Prop. 3.

unreal We also maintain a predicate $\mathsf{unreal}(\mathbb{A})$, which expresses the fact that $\mathbb{A}$ cannot ever be realized, i.e. there is no $H \colon \mathbb{A} \mapsto \mathbb{A}'$ for any realized $\mathbb{A}'$.

There are three main facts about unreal:

Proposition 9. *1. If* $a \in \mathsf{non}_{\mathbb{A}}$ *and* $(\mathsf{Lsn}[a]) \downarrow 1 \in \mathbb{A}$*, then* $\mathsf{unreal}(\mathbb{A})$*.*
2. If $\neg\mathsf{realized}(\mathbb{A})$*, and, moreover,* $\mathsf{unreal}(\mathbb{A}')$ *for all* $\mathbb{A}'$ *such that* $\mathsf{step}(\mathbb{A}, \mathbb{A}')$*, then* $\mathsf{unreal}(\mathbb{A})$*.*
3. $\mathsf{unreal}(\mathbb{A}')$ *and* $H\colon \mathbb{A}' \mapsto \mathbb{A}''$ *implies* $\mathsf{unreal}(\mathbb{A}'')$*.*

Clause 2 says that, for unrealized skeletons, unreal propagates backward through step. Clause 3 says that unreal is preserved under homomorphisms.

Given a starting skeleton $\mathbb{A}_0$, we would like to determine the set

$$\{\mathbb{A}_2\colon \exists \mathbb{A}_1 \,.\, \mathsf{step}^*(\mathbb{A}_0, \mathbb{A}_1) \ \wedge \ \mathsf{shape}(\mathbb{A}_1, \mathbb{A}_2)\}.$$

To do so, we use the following search algorithm.

At any stage, we have a skeleton of interest $\mathbb{A}$, and we know part of the extensions of the relations step and unreal. We also maintain a set $\mathcal{F}$, the "fringe" of skeletons we have encountered but not yet taken a step from.

Initially, $\mathbb{A}$ is typically a skeleton containing a single strand of the role whose "point of view" is under analysis, for instance the right side of Fig. 1, 2, or 3. $\mathbb{A}$ has a choice of $\mathsf{non}_{\mathbb{A}}, \mathsf{unique}_{\mathbb{A}}$ indicating the assumptions under which we would like to analyze the protocol. Initially, $\mathcal{F} = \emptyset$.

1. Take cases on $\mathbb{A}$.
   $\mathbb{A}$ **is realized.** If $\mathbb{A}$ is realized, we extract one or more shapes $\mathbb{A}'$ (Prop. 3), recording $\mathsf{shape}(\mathbb{A}, \mathbb{A}')$. Continue with step 2.
   $\mathbb{A}$ **may be pruned.** If $\mathbb{A}$ contains a redundant copy of a strand, then we discard $\mathbb{A}$ (Prop. 8). Continue with step 2.
   $\mathbb{A}$ **contains an unsatisfied test.** If $\mathbb{A}$ is unrealized, then we look for either an outgoing test pair or else an incoming test node without any transforming node. Applying Theorem 1, or respectively Theorem 2, we obtain $k \geq 0$ candidate augmentations, contractions, and listeners, yielding the skeletons $\mathbb{A}_1 \ldots, \mathbb{A}_k$.
   If an outgoing test pair or incoming test node is found, but $k = 0$, then $\mathbb{A}$ is unrealizable; record $\mathsf{unreal}(\mathbb{A})$ and use Prop. 9 to propagate unreal backward through step for unrealized skeletons.
   If $k \geq 1$, record $\mathsf{step}(\mathbb{A}, \mathbb{A}_i)$ for each i with $1 \leq i \leq k$. Letting S be the subset of $\{\mathbb{A}_1 \ldots, \mathbb{A}_k\}$ that have not yet been considered, then the new $\mathcal{F}$ is the old $\mathcal{F} \cup S$. Continue with step 2.
2. Now prune $\mathcal{F}$.
   Apply Prop. 9 to $\mathcal{F}$: If any $\mathbb{A}' \in \mathcal{F}$ has a listener strand $\mathsf{Lsn}[a]$ with $a \in \mathsf{non}_{\mathbb{A}'}$, then record $\mathsf{unreal}(\mathbb{A}')$ and propagate unreal backward through step for unrealized skeletons. We discard any skeletons $\mathbb{A}'$ such that unreal from $\mathcal{F}$.
3. Now select the next skeleton $\mathbb{A}$ from $\mathcal{F}$ if possible. If $\mathcal{F} = \emptyset$, then exit.
   If for some $\mathbb{A}'$, $\mathcal{F} = \{\mathbb{A}'\} \cup \mathcal{F}'$, where $\mathbb{A}' \notin \mathcal{F}'$, then let $\mathbb{A} = \mathbb{A}'$ and $\mathcal{F} = \mathcal{F}'$, and go to step 1.

By Prop. 7, in step 1, at least one of the cases applies. In particular, if $\mathbb{A}$ is not realized, then there is an unsatisfied test in $\mathbb{A}$.

Theorems 1 and 2 justify one characteristic of this search algorithm. Namely, a skeleton $\mathbb{A}$ is considered at most once, and only one incoming or outgoing transformed edge is considered. The propositions say that any homomorphism to a realized skeleton can still be found, even if this transformed edge is solved first. **Clarify choice of $\mathbb{A}_1 \ldots, \mathbb{A}_k$.**

7 Shape Analysis of Yahalom's Protocol

We return now to Yahalom's protocol. We would like to find the shapes accessible from an initial skeleton that contains all four nodes of a strand s_r of responder B, using parameters A, B, N_a, N_b, K. We assume the long term keys $\mathsf{ltk}(A), \mathsf{ltk}(B)$ uncompromised, and N_b freshly chosen. Let $\mathsf{N}_0 = \{s_r \downarrow i \colon 1 \leq i \leq 4\}$, and let $\preceq_0$ be the strand ordering on s. Starting from the initial skeleton

$$\mathbb{A}_0 \;=\; (\mathsf{N}_0,\; \preceq_0,\; \{\mathsf{ltk}(A), \mathsf{ltk}(B)\},\; \{N_b\}),$$

we would like to find all accessible shapes.

The initiator strand. The pair of nodes $((s_r \downarrow 2),\ (s_r \downarrow 4))$ is an outgoing test pair for N_b, relative to the set of terms $S_1 =$

$$\begin{array}{rl} & \{\, \{\!|A \mathbin{\hat{}} N_a \mathbin{\hat{}} N_b|\!\}_{\mathsf{ltk}(B)} \,\} \\ \cup & \{\, \{\!|B \mathbin{\hat{}} K' \mathbin{\hat{}} N_a \mathbin{\hat{}} N_b|\!\}_{\mathsf{ltk}(A)} \;:\; K' \in \mathrm{keys} \,\} \end{array}$$

and the set of keys $X_0 = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$. Since $\mathsf{ltk}(A), \mathsf{ltk}(B) \in n_{\mathbb{A}_0}$, when we apply Theorem 1, the skeletons resulting from Clause 3 are in unreal (Prop. 9). Thus, for homomorphisms H that do not contract parameters of s_r, there is only one possibility: H adds an outgoing transforming edge $m_0 \Rightarrow^+ m_1$. In $\mathrm{term}(m_0)$, N_b occurs only within S_1, but N_b occurs outside S_1 in $\mathrm{term}(m_1)$.

The structure of the protocol in Fig. 3, provides only one possibility: $m_0 \Rightarrow^+ m_1$ lies on an instance s_i of the initiator role, specifically $(s_i \downarrow 2) \Rightarrow (s_i \downarrow 3)$. Matching $\mathrm{term}(s_i \downarrow 2)$ with $\{\!|B \mathbin{\hat{}} K' \mathbin{\hat{}} N_a \mathbin{\hat{}} N_b|\!\}_{\mathsf{ltk}(A)}$, we infer s_i is an initiator strand with A, B, N_a, N_b and some session key K'. In the resulting skeleton $\mathbb{A}_1$ (Fig. 7), $(s_r \downarrow 2) \preceq_{\mathbb{A}_1} (s_i \downarrow 2)$ and $(s_i \downarrow 3) \preceq_{\mathbb{A}_1} (s_r \downarrow 4)$. $\mathbb{A}_1$'s non-originating and uniquely originating atoms are unchanged.

The server strand. We can now introduce the server strand using either an outgoing augmentation or an incoming augmentation using $s_i \downarrow 2$. We prefer the latter, since it is simpler.

Since $\mathrm{term}(s_i \downarrow 2) = \{\!|B \mathbin{\hat{}} K' \mathbin{\hat{}} N_a \mathbin{\hat{}} N_b|\!\}_{\mathsf{ltk}(A)}$ and $\mathsf{ltk}(A) \in n_{\mathbb{A}_0}$, the listener strand is unrealizable. Thus, there is a regular node m_1 that precedes $s_i \downarrow 2$ and transmits $\mathrm{term}(s_i \downarrow 2)$. The structure of the protocol in Fig. 3, provides only one possibility: m_1 is node 2 of a server strand s_s. Reading the parameters off $\mathrm{term}(m_1)$, the parameters of s_s are A, B, N_a, N_b, K'. The resulting skeleton $\mathbb{A}_2$ (Fig. 8) has unchanged non-originating atoms. However, the strand s_s freshly generates K', i.e. $u_{s_s} = \{K'\}$, so $u_{\mathbb{A}_2} = \{N_b, K'\}$.

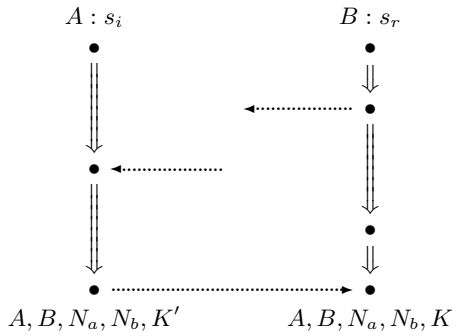

Fig. 7. Skeleton $\mathbb{A}_1$, with $n_{\mathbb{A}_1} = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$, $u_{\mathbb{A}_1} = \{N_b\}$

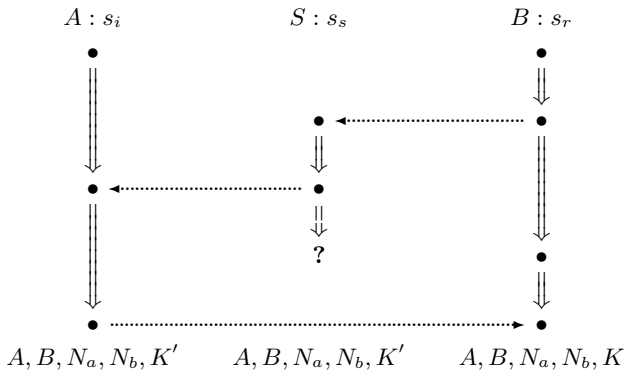

Fig. 8. Skeleton $\mathbb{A}_2$, with $n_{\mathbb{A}_2} = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$, $u_{\mathbb{A}_2} = \{N_b, K'\}$

Does $K' = K$***?*** We apply the outgoing test with $(n_0, n_1) = ((s_r \downarrow 2), (s_r \downarrow 4))$, using the set $S_2 = S_1 \cup \{\{\!|N_b|\!\}_{K'}\}$. Here we have three cases: (1) Eliminate the transformed edge with a contraction; (2) Find an outgoing augmentation; or (3) Add a listener strand for K'.

Case 1: Contract K' ***to*** K***.*** The mapping $\alpha = [K' \mapsto K]$ destroys the outgoing test edge, since $S_2 \cdot \alpha$ contains $\{\!|N_b|\!\}_K$. Let $\mathbb{A}_3 = \mathsf{hull}_\alpha(\mathbb{A}_2)$.

Case 2: Find an outgoing augmentation. An initiator strand s_i' with parameters A, B, N_a, N_b, K'', for $K'' \neq K$, would receive N_b only within S_2 and transmit it in the form $\{\!|N_b|\!\}_{K''}$, so that N_b no longer occurs only within S_2. However, s_i' is a copy of s_i, in the sense of Prop. 8, so that we will prune the resulting skeleton.

Case 3: Add a listener strand $\mathsf{Lsn}[K']$***.*** We add a listener strand $s_\ell = \mathsf{Lsn}[K']$ to $\mathbb{A}_2$, and now have an outgoing transformed edge with $(n_0, n_1) = s_s \downarrow 2, s_\ell \downarrow 1$, where the uniquely originating parameter is K', the set $S_3 = \{\{\!|B \hat{} K' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}, \{\!|A \hat{} K'|\!\}_{\mathsf{ltk}(B)}\}$, and $X = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$. No contraction will eliminate this transformed edge; no outgoing augmentation will resolve it; and no listener strand for a member of X can be realized. So the listener strand s_ℓ is not realizable.

Thus, only the contraction $\alpha = [K' \mapsto K]$ can lead to a live skeleton, $\mathbb{A}_3$.

The source of $s_r \downarrow 3$***.*** Finally, $s_r \downarrow 3$ is an incoming test node in $\mathbb{A}_3$, which because $\mathsf{ltk}(B) \in n_{\mathbb{A}_3}$, can be resolved only by an incoming augmentation. Again by the form of the protocol, only a server strand s_s' can be the origin of $\{\!|A \hat{} K|\!\}_{\mathsf{ltk}(B)}$. Thus, s_s' uses parameters A, B, K, although the nonces are not yet clear. However, since K originates uniquely in $\mathbb{A}_3$, and originates on both s_s and s_s', we may identify $s_s = s_s'$, thus resolving the remaining parameters. The resulting skeleton $\mathbb{A}_4$ is realized, and is shown in Fig. 9. We may check that no common occurrences of the same atom can be distinguished while preserving realization. Thus, $\mathbb{A}_4$ is a shape for $\mathbb{A}_0$. We have no remaining skeletons to explore, and so our search has terminated with just one skeleton.

8 Implementing CPSA

In implementing CPSA, we have represented terms, strands, and skeletons in straightforward ways. A homomorphism $[\phi, \alpha] \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ is also represented directly via an association of strands in $\mathbb{A}_0$ to strands in $\mathbb{A}_1$ (for ϕ), together with a type-respecting association of atoms to atoms (for the replacement α). When CPSA parses a protocol description Π, it identifies the nodes that are potential transforming nodes for incoming tests, and the potential transforming edges $m_0 \Rightarrow^+ m_1$ for outgoing tests.

CPSA annotates skeletons with additional information about safe atoms. We say that a is a *safe atom* in $\mathbb{A}$ if $\mathsf{unreal}(\mathbb{A}')$ when $\mathbb{A}'$ is $\mathbb{A}$ augmented with $\mathsf{Lsn}[a]$.

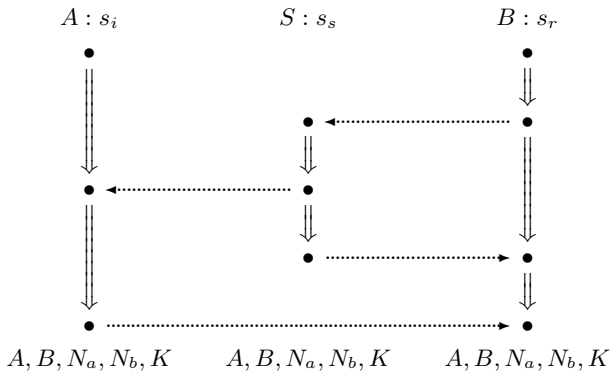

Fig. 9. Skeleton $\mathbb{A}_4$, with $n_{\mathbb{A}_4} = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$, $u_{\mathbb{A}_4} = \{N_b, K\}$

By Prop. 9, Clause 3, safe atoms are preserved under homomorphisms. I.e., if $[\phi, \alpha] \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$, and a is a safe atom in $\mathbb{A}_0$, then $a \cdot \alpha$ is a safe atom in $\mathbb{A}_1$. Safe atoms observed early in a search may be used repeatedly to prune many dead branches later in the search. For this reason, we explore the listener augmentation branches in our search before outgoing and incoming augmentations.

A tricky matter was how to select the sets S to use in applications of Theorem 1. We settled on a trick we call the "forwards-then-backwards" technique. CPSA plans a sequence of applications of Theorem 1, like Steps 1 and 2 of Section 7. To do so, it follows the transmission of the uniquely originating value—N_b in that case—forwards. Essentially, it follows the intuitive, forwards order of Steps 1–3 in Section 2. Since Step 3 does not produce an outgoing augmentation, but only a contraction, it is deferred; each of the previous steps has suggested an S. We have

1. $S_1 = \{\{\!|A \mathbin{\hat{}} N_a \mathbin{\hat{}} N_b|\!\}_{\mathsf{ltk}(B)}\}$, and
2. $S_2 = S_1 \cup \{\{\!|B \mathbin{\hat{}} K' \mathbin{\hat{}} N_a \mathbin{\hat{}} N_b|\!\}_{\mathsf{ltk}(A)} \colon K' \text{ is a key } \}$

CPSA uses the sets in the opposite order, i.e. S_2 is used first to introduce the initiator strand s_i in Step 1 of Section 7, and then S_1 is used to introduce the first two nodes of the server strand s_s. These occur earlier in the resulting skeleton than the transforming edge introduced using S_2. The forwards-then-backwards technique appears to produce a "good enough" selection of sets S to preserve the completeness property, Corollary 3, but we have not carefully established this.

The forwards-then-backwards technique also suggested CPSA's representation for the sets S. These sets are not necessarily finite; S_2 for instance is not. However, they are generated by the operations of union and set difference from sets that are either

1. singletons $\{t\}$ or
2. all the instances of $\{t\}$ obtained as some of its parameters vary.

For instance, $\{\{|B \hat{} K' \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(A)} : K' \text{ is a key } \}$ consists of all instances of $\{|B \hat{} K' \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(A)}$ obtained as parameter K' varies. Of course, the singleton t is the set of all instance of t obtained as none of its parameters vary. Thus, we represent the sets S as finite unions and differences of values $\lambda \boldsymbol{v} \,.\, t$, where the vector $\boldsymbol{v}$ binds 0 or more atoms occurring in t. Restricting S to the sets representable in this form does not falsify the completeness property, as can be seen from the proof of Prop. 7.

In practice, we use unification to determine relevant contractions. We also use unification to combine augmentation and contraction steps. For instance, if $\mathbb{A}$ contains an outgoing test relative to a, S, X, we unify the transforming edges $m_0 \Rightarrow^+ m_1$ in the protocol Π with the components $\lambda \boldsymbol{v} \,.\, t$ in the representation of S. A most general unifier α is successful if $a \cdot \alpha$ occurs only within $S \cdot \alpha$ in $\mathrm{term}(m_0 \cdot \alpha)$, but it occurs outside $S \cdot \alpha$ in $\mathrm{term}(m_1 \cdot \alpha)$. When this is the case, we obtain the new skeleton $\mathbb{A} \cdot \alpha$ augmented with $r \cdot \alpha$, where $r \in \Pi$ is the role that $m_0 \Rightarrow^+ m_1$ occurs on. In practice, this provides an extremely focused search. This use of unification preserves the completeness property. Some protocols and runtimes on a Thinkpad X31, with a 1.4 GHz Pentium M processor and 1 GB store, running Linux, are shown in Fig. 10. CPSA is implemented in OCaml.

Protocol	Point of view	Runtime
ISO reject	responder	0.193s
Kerberos	client	1.443s
Needham-Schroeder	responder	0.055s
Needham-Schroeder-Lowe	responder	0.124s
Yahalom	responder	2.709s

Fig. 10. Protocols with CPSA runtimes

In future work, we intend to augment CPSA with keys generated by hashing complex terms, which many protocols use for key derivation. We also plan to incorporate Diffie-Hellman operations on atoms, as studied in [11]. We also believe that it will be straightforward to lift the atoms-to-atoms restriction on replacement, and obtain a framework in which compound terms may be instantiated for parameters. We preferred, however, to work out the underlying theory in the simpler atoms-to-atoms context first.

References

1. Bruno Blanchet and Andreas Podelski. Verification of cryptographic protocols: Tagging enforces termination. In Andrew D. Gordon, editor, *Foundations of Software Science and Computation Structures*, number 2620 in LNCS, pages 136–152. Springer, April 2003.
2. Michele Boreale. Symbolic trace analysis of cryptographic protocols. In *ICALP*, 2001.

3. Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society*, Series A, 426(1871):233–271, December 1989.
4. Shaddin Doghmi, Joshua Guttman, and F. Javier Thayer. Skeletons and the shapes of bundles. Technical report, The MITRE Corp., 2005. Available at `http://www.ccs.neu.edu/home/guttman/skeletons.pdf`.
5. Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Skeletons, homomorphisms, and shapes: Characterizing protocol executions. Submitted for publication, June 2006.
6. Nancy Durgin, Patrick Lincoln, John Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004. Initial version appeared in *Workshop on Formal Methods and Security Protocols*, 1999.
7. Marcelo Fiore and Martín Abadi. Computing symbolic models for verifying cryptographic protocols. In *Computer Security Foundations Workshop*, June 2001.
8. Joshua D. Guttman. Key compromise and the authentication tests. *Electronic Notes in Theoretical Computer Science*, 47, 2001. Editor, M. Mislove. URL `http://www.elsevier.nl/locate/entcs/volume47.html`, 21 pages.
9. Joshua D. Guttman and F. Javier Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 283(2):333–380, June 2002.
10. 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.
11. Jonathan C. Herzog. The Diffie-Hellman key-agreement scheme in the strand-space model. In *16th Computer Security Foundations Workshop*, pages 234–247, Asilomar, CA, June 2003. IEEE CS Press.
12. Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceeedings of* TACAS, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer Verlag, 1996.
13. Gavin Lowe. Casper: A compiler for the analysis of security protocols. In *10th Computer Security Foundations Workshop Proceedings*, pages 18–30. IEEE Computer Society Press, 1997.
14. Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *8th ACM Conference on Computer and Communications Security (CCS '01)*, pages 166–175. ACM, 2001.
15. Lawrence C. Paulson. Mechanized proofs of a recursive authentication protocol. In *10th IEEE Computer Security Foundations Workshop*, pages 84–94. IEEE Computer Society Press, 1997.
16. Adrian Perrig and Dawn Xiaodong Song. Looking for diamonds in the desert: Extending automatic protocol generation to three-party authentication and key agreement protocols. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, July 2000.
17. R. Ramanujam and S. P. Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13(1):135–166, 2005. Preliminary version appeared in WITS '03, *Workshop on Issues in the Theory of Security*, Warsaw, April 2003.
18. Peter Y. A. Ryan and Steve Schneider. An attack on APM's recursive authentication protocol—a cautionary tale. *Information Processing Letters*, 65, 1998.

A Unifying Equivalence Relations

The essential content of Prop. 2 is that there is a coarsest equivalence relation that satisfies the cascading conditions mentioned in Section 4.2.

We will consider equivalence relations on atomic terms which are type preserving. This means $a \mathbf{R} b$ implies a, b have the same type.

The set of equivalence relations $\mathbf{R}$ on atomic terms forms a partially ordered set with respect to coarsening $\sqsubseteq$. Thus is $\mathbf{R} \sqsubseteq \mathbf{S}$ iff $\mathbf{R}$ is a coarsening of $\mathbf{S}$, or equivalently $\mathbf{R} \subseteq \mathbf{S}$. Note that $\mathbf{R} \sqsubseteq \mathbf{S}$ iff the each equivalence class of $\mathbf{R}$ is a subset of some equivalence class of $\mathbf{S}$. Thus $\mathbf{R} \sqsubseteq \mathbf{S}$ implies the equivalence classes of $\mathbf{S}$ form a coarsening of those of $\mathbf{R}$. The two extremes are the equality relation and the relation which identifies everything.

Equiv with coarsening is a lattice $(\mathbf{Equiv}, \sqsubseteq)$.

An equivalence relation $\mathbf{R}$ on atoms induces an equivalence relation also denoted $\mathbf{R}$ on terms, defined by recursion as follows. For non-atomic terms s, t $s \mathbf{R} t$ iff t and s both have the same constructor and the components $s_1, \ldots s_n$ of s and $t_1, \ldots t_n$ of t, s satisfy $s_i \mathbf{R} t_i$ for $i = 1, \ldots, n$.

Proposition 10. *If $\mathbf{R}_\lambda$ is a family of equivalence relations on atoms, and s, t are terms then $s \big[\bigcap_\lambda \mathbf{R}_\lambda\big] t$ iff $s \mathbf{R}_\lambda t$ for all λ.*

Proof. Structural induction. If s, t are atomic this is by definition. Otherwise, $s = \mathbf{F}(s_1, \ldots s_n)$, $t = \mathbf{F}(t_1, \ldots t_n)$. Then

$$\begin{aligned} s\big[\bigcap_\lambda \mathbf{R}_\lambda\big] t &\iff s_i \big[\bigcap_\lambda \mathbf{R}_\lambda\big] t_i \\ &\iff \forall \lambda, i = 1, \ldots, n, s_i \mathbf{R}_\lambda t_i \\ &\iff \forall \lambda, s \mathbf{R}_\lambda t. \end{aligned}$$

If S is a set of terms, the unification of S, denoted $\mathbf{Unif}(S)$ is the smallest equivalence $\mathbf{R}$ relation under which all elements of S are equivalent with respect to $\mathbf{R}$.

Definition 15. Let $\mathbb{A}$ be a pre-skeleton. Strands s, s' are equivalent with respect to an equivalence relation $\mathbf{R}$ iff for every index k, if the nodes $s \downarrow k$ and $s' \downarrow k$ are both defined, then they have the same sign and $\mathrm{term}(s \downarrow k) \mathbf{R} \mathrm{term}(s' \downarrow k)$.

Nodes $n = s \downarrow k$, $n' = s' \downarrow k'$ are equivalent iff $s \mathbf{R} s'$ and $k = k'$.

In the above definition, s and s' are not required to have the same length.

A transitive relation $\preceq$ on a set X is *invariant* under an equivalence relation $\equiv$ iff $m \preceq n$, $m \equiv m'$ and $n \equiv n'$ imply $m' \preceq n'$. The invariance property implies that the relation $m \preceq n$ is determined by the $\equiv$ classes of m and n respectively, and thus passes to the quotient space $X/\equiv$.

If $\equiv$ is an equivalence relation on X and $\preceq$ is a transitive relation on X, define

$$m \preceq' n \iff \exists m_1, n_1, \cdots m_k, n_k \text{ s.t } m = m_1 \preceq n_1 \equiv m_2 \preceq n_2 \equiv m_3 \cdots m_k \preceq n_k = n.$$

Then $\preceq'$ is the coarsest transitive refinement of $\preceq$ on X invariant under $\equiv$.

Definition 16. An equivalence relation $\mathbf{R}$ on atoms is order compatible with the pre-skeleton $\mathbb{A}$ iff whenever $m_1 \preceq_{\mathbb{A}} n_1 \,\mathbf{R}\, m_2 \preceq_{\mathbb{A}} n_2 \,\mathbf{R}\, m_3 \cdots m_k \preceq_{\mathbb{A}} n_k \,\mathbf{R}\, m_1$ then, for all i, $m_i \,\mathbf{R}\, n_i$.

Lemma 17. *An equivalence relation* $\mathbf{R}$ *on atoms is order compatible with a pre-skeleton* $\mathbb{A}$ *iff the coarsest* $\mathbf{R}$ *invariant refinement* $\preceq'$ *of* $\preceq_{\mathbb{A}}$ *has the property that if* $m \preceq' n$ *and* $n \preceq' m$*, then* $m \,\mathbf{R}\, n$*.*

It follows from the previous lemma that given an $\mathbb{A}$-order compatible equivalence relation $\mathbf{R}$, the quotient structure obtained by collapsing $\mathbf{R}$ equivalent nodes of $\mathbb{A}$, equipped with the image order structure is a pre-skeleton $\mathbb{A}/\mathbf{R}$, and the map $\mathbb{A} \to \mathbb{A}/\mathbf{R}$ is a homomorphism.

Proposition 11. *If* $\mathbf{R}_\lambda$ *is a family of equivalence relations on atoms which are order compatible with the pre-skeleton* $\mathbb{A}$*, then* $\bigcap_\lambda \mathbf{R}_\lambda$ *is order compatible with* $\mathbb{A}$*.*

Definition 18. An equivalence relation $\mathbf{R}$ on atoms *collapses* a pre-skeleton $\mathbb{A}$ iff for every atom a, the set of strands

$$\{s : \text{for some atom } b,\ b \,\mathbf{R}\, a \text{ and } b \text{ originates on } s\}$$

is empty if $a \in \mathsf{non}_{\mathbb{A}}$ and consists of $\mathbf{R}$ equivalent strands if $a \in \mathsf{unique}_{\mathbb{A}}$.

Proposition 12. *If* $\mathbf{R}_\lambda$ *is a family of equivalence relations on atoms each of which collapses the pre-skeleton* $\mathbb{A}$*, then* $\bigcap_\lambda \mathbf{R}_\lambda$ *collapses* $\mathbb{A}$*.*

Proposition 13. *If* $\mathbf{R}$ *is an equivalence relation on atoms which has a compatible and collapsing refinement, then the there is a coarsest compatible collapsing refinement.*