SEARCHING FOR SHAPES IN CRYPTOGRAPHIC PROTOCOLS (EXTENDED VERSION)

SHADDIN F. DOGHMI, JOSHUA D. GUTTMAN, AND F. JAVIER THAYER

Abstract. We describe a method for enumerating all essentially different executions possible for a cryptographic protocol. We call them the *shapes* of the protocol. Naturally occurring protocols have only finitely many, indeed very few shapes. Authentication and secrecy properties are easy to determine from them, as are attacks. cpsa, our Cryptographic Protocol Shape Analyzer, implements the method.

In searching for shapes, cpsa starts with some initial behavior, and discovers what shapes are compatible with it. Normally, the initial behavior is the point of view of one participant. The analysis reveals what the other principals must have done, given this participant's view.

1. Introduction

The executions of cryptographic protocols frequently have very few essentially different 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. We may also find 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 complete method for enumerating the shapes of a protocol within a pure Dolev-Yao model. If the protocol has only finitely many essentially different shapes, the enumeration will terminate. From the shapes, we can then read off the answers to secrecy and authentication questions and observe other anomalies. Our software implementation of this method is called a Cryptographic Protocol Shapes Analyzer (cpsa).

We use the strand space theory [8]. A *skeleton* [4] represents regular (non-penetrator) behavior that might make up part of an execution, and a *homomorphism* is an information-preserving map between skeletons. Skeletons are partially-ordered structures, like fragments of Lamport diagrams [10]. A skeleton is *realized* if it is non-fragmentary, i.e. it contains exactly the regular behavior of some execution. A realized skeleton is a *shape* if it is minimal in a sense we will make precise. We *search* for shapes using the authentication tests [8] to find new strands to add when a skeleton is not large enough to be realized.

The main technical result underlying cpsa is *completeness*, in the sense that—for any protocol—our authentication test search eventually discovers every shape for that protocol (Thm. 6.10). It cannot terminate for every protocol [5]. It does, however, terminate for reasonably inclusive classes [2, 16].

[1]Supported by the National Security Agency and by MITRE-Sponsored Research. Addresses: shaddin@stanford.edu, {guttman, jt}@mitre.org.

CPSA's search is related to the second version of Athena [15], which adopted the authentication tests from early versions of [8]. Our work, however, is distinguished from Athena in several ways. First, it involves the regular behaviors alone; we never represent adversary activity within a shape. Second, we have improved both the search and the theory. In particular, we have introduced the notion of shape, which defines a criterion for which possible executions should be considered, among the infinitely many executions (of unbounded size) of any protocol. Third, we have now created versions of the authentication tests strong enough for completeness to be true.

The type-and-effect system for spi calculus [6] is also related to the authentication tests, but differs from our work in two ways. First, we do not use the syntactically-driven form of a type system, but instead a direct analysis of behaviors. Second, type-and-effect systems aim at a sound approximation, whereas our work provides actual counterexamples when a security goal is not met.

The shapes describe protocol executions of all sizes; we do not follow the widely practiced *bounded* protocol analysis (e.g. [1, 12]).

Structure of this paper. We will develop the CPSA search strategy starting with examples. Section 2 contains a very brief example of a protocol and its shapes, and introduces terminology.

Section 3 introduces the Yahalom protocol [3, 14], and examines what global behavior must be present in a run that contains a given local behavior. This protocol is a good example, because although it is quite compact, it illustrates almost every aspect of the CPSA search method.

We expand the terminology and give more precise definitions in Section 4 and, for skeletons, homomorphisms, and shapes, Section 5. In this we use Section 3 as a source of examples. Section 6 develops the authentication test theorems that were used repeatedly—in implicit form—in Section 3.

In Section 7, we explain the search algorithm that the authentication tests suggest, as illustrated within Section 3. First, in Section 7.1, we extract the search content of the authentication test theorems themselves, abstracting the two principles used repeatedly (Section 7.1). In Section 7.2, we define the search's control structure. The CPSA implementation is the subject of Section 8.

2. A Small Example with the Core Ideas

In practice, protocols have remarkably few shapes. The Needham-Schoeder-Lowe [13, 11] protocol has only one. This holds whether we take the point of view of the responder B, asking what behavior must have occurred if B has had a full local run of the protocol, or whether we take the point of view of the originator A. In either case, the other party must have had a matching run. The initiator, however, can never be sure that the last message it sends was received by the responder, as it is no longer expecting to receive any further messages.

Uniqueness of shape is perhaps not surprising for as strong a protocol as Needham-Schroeder-Lowe. However, even a flawed protocol such as the original Needham-Schroeder protocol may have a unique shape, shown in Fig. 1.

2.1. **Terminology.** B's behavior is represented by the right-hand column in Fig. 1, consisting of nodes connected by double arrows $\bullet \Rightarrow \bullet$. A's behavior is represented by the left-hand column. We call such a column a *strand*. The *nodes* represent

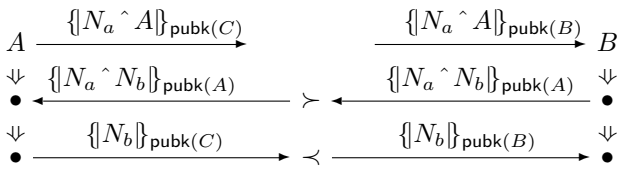

FIGURE 1. Needham-Schroeder Shape ($\mathsf{privk}(A)$ uncompromised, N_b fresh, B's point of view)

message transmission or reception events, and the double arrows represent succession within a single linearly ordered local activity. The message transmitted or received on a node n is written $\mathsf{msg}(n)$. A *regular strand* is a strand that represents a principal executing a single local session of a protocol; it is called a regular strand because the behavior follows the protocol rules.

In the messages, we use $\{|t|\}_K$ to refer to the encryption of t with key K, and $t \hat{} t'$ means the pair of the messages t and t'. Messages are constructed freely via these two operations from atomic values A, N_a, K, etc.

The *subterm* relation is the least reflexive, transitive relation such that t is a subterm of $\{|t|\}_K$, t is a subterm of $t \hat{} t'$, and t is a subterm of $t' \hat{} t$ (for all K, t'). We write $t \sqsubseteq t'$ if t is a subterm of t'. Thus, $K \not\sqsubseteq \{|t|\}_K$ unless (anomalously) $K \sqsubseteq t$. Instead, K contributed to *how* $\{|t|\}_K$ was produced. This terminology has an advantage: Uncompromised long-term keys are never subterms of messages transmitted in a protocol; they are used by regular principals to encrypt, decrypt, or sign messages, but are never transmitted. A value a *originates at* a node n if (1) n is a transmission node; (2) $a \sqsubseteq \mathsf{msg}(n)$; and (3) if m is any earlier node on the same strand, then $a \not\sqsubseteq \mathsf{msg}(m)$.

Adversary behavior is represented by strands too. These *penetrator strands* codify the basic abilities that make up the Dolev-Yao model. They include transmitting a basic value such as a nonce or a key; transmitting an encrypted message after receiving its plaintext and the key; and transmitting a plaintext after receiving ciphertext and decryption key. The adversary can also pair two messages, or separate the pieces of a paired message. Since a penetrator strand that encrypts or decrypts must receive the key as one of its inputs, keys used by the adversary—compromised keys—have always been transmitted by some participant. These penetrator strands are independent of the protocol under analysis.

2.2. **The NS Shape.** Suppose B's nonce N_b has been freshly chosen and A's private key $\mathsf{privk}(A)$ is uncompromised, and B has had the full run shown at the right in Fig. 1. Given that—on a particular occasion—B has received and sent these messages in this order, what else must have occurred elsewhere in the network?

A must have had a matching run, with the messages sent and received in the order indicated by the arrows of both kinds and the connecting dots. The dots mean that the endpoints are ordered, but that other behavior may intervene, whether adversary behavior or regular strands. There is no alternative: Any diagram containing the responder strand of Fig. 1 must at least contain the initiator strand, with the events ordered as shown, or it cannot have happened.

Such a diagram is a *shape*. A shape consists of the regular strands of an execution, forming a *minimal* set among executions containing certain regular strands (in this

case, just the right-hand column). Possible execution may freely add adversary behavior. Each shape is relative to assumptions about keys and freshness, in this case that $\mathsf{privk}(A)$ is uncompromised and N_b freshly chosen.

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 [11]; or else (2) $C = B$, leading to the intended run.

Some protocols have more than one shape, Otway-Rees, e.g., having four. In searching for shapes, one starts from some initial set of strands. Typically, the initial set is a singleton, which we refer to as the "point of view" of the analysis.

2.3. **Skeletons, Homomorphisms, Shapes.** A *skeleton* $\mathbb{A}$ is (1) a finite set of regular nodes, equipped with additional information. The additional information consists of (2) a partial order $\preceq_{\mathbb{A}}$ on the nodes indicating causal precedence; (3) a set of keys $\mathsf{non}_{\mathbb{A}}$; and (4) a set of atomic values $\mathsf{unique}_{\mathbb{A}}$. Values in $\mathsf{non}_{\mathbb{A}}$ must originate nowhere in $\mathbb{A}$, whereas those in $\mathsf{unique}_{\mathbb{A}}$ originate at most once in $\mathbb{A}$.[1] $\mathbb{A}$ is *realized* if it has precisely the regular behavior of some concrete execution. Every message received by a regular participant either should have been sent previously, or should be constructable by the adversary with the help of messages sent previously. Fig. 1 shows a skeleton, indeed a realized one. See Defs. 5.1, 5.2.

A *homomorphism* is a map H from $\mathbb{A}_0$ to $\mathbb{A}_1$, written $H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$. We represent it as a pair of maps (ϕ, α), where ϕ maps the nodes of $\mathbb{A}_0$ into those of $\mathbb{A}_1$, and α is a *replacement* mapping atomic values into atomic values. We write $t \cdot \alpha$ for the result of applying a replacement α to a message t. $H = (\phi, \alpha)$ is a homomorphism iff: (1) ϕ respects strand structure, and $\mathsf{msg}(n) \cdot \alpha = \mathsf{msg}(\phi(n))$ for all $n \in \mathbb{A}_0$; (2) $m \preceq_{\mathbb{A}_0} n$ implies $\phi(m) \preceq_{\mathbb{A}_1} \phi(n)$; (3) $\mathsf{non}_{\mathbb{A}_0} \cdot \alpha \subset \mathsf{non}_{\mathbb{A}_1}$; and (4) $\mathsf{unique}_{\mathbb{A}_0} \cdot \alpha \subset \mathsf{unique}_{\mathbb{A}_1}$. See Def. 5.3.

Homomorphisms are *information-preserving* transformations. Each skeleton $\mathbb{A}_0$ describes the realized skeletons reachable from $\mathbb{A}_0$ by homomorphisms. Since homomorphisms compose, if $H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ then any realized skeleton accessible from $\mathbb{A}_1$ is accessible from $\mathbb{A}_0$. Thus, $\mathbb{A}_1$ preserves the information in $\mathbb{A}_0$: $\mathbb{A}_1$ describes a subset of the realized skeletons described by $\mathbb{A}_0$.

A homomorphism may supplement the strands of $\mathbb{A}_0$ with additional behavior in $\mathbb{A}_1$; it may affect atomic parameter values; and it may identify different nodes together, if their strands are compatible in messages sent and positions in the partial ordering. For instance, a map is a homomorphism if it embeds a single strand of Fig. 1 (e.g. B's strand on the right side) into the whole skeleton shown. Likewise if we embed the first two nodes of B's strand (rather than the whole sequence of three) into the whole of Fig. 1. Another homomorphism rewrites each occurrence of C in Fig. 1 to B, hence each occurrence of $\mathsf{pubk}(C)$ to $\mathsf{pubk}(B)$. It yields the Needham-Schroeder intended run.

A homomorphism $H = (\phi, \alpha)$ is *nodewise injective* if ϕ is an injective function on the nodes. The nodewise injective homomorphisms determine a useful partial order on homomorphisms: When for some nodewise injective H_1, $H' = H_1 \circ H$, we write $H \leq_n H'$. If $H \leq_n H' \leq_n H$, then H and H' are isomorphic.

A homomorphism $H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ is a *shape* iff (a) $\mathbb{A}_1$ is realized and (b) H is $\leq_n$-minimal among homomorphisms from $\mathbb{A}_0$ to realized skeletons.

[1] When $n \Rightarrow^* n'$ and $n' \in \mathbb{A}$, we require $n \in \mathbb{A}$ and $n \preceq_{\mathbb{A}} n'$.

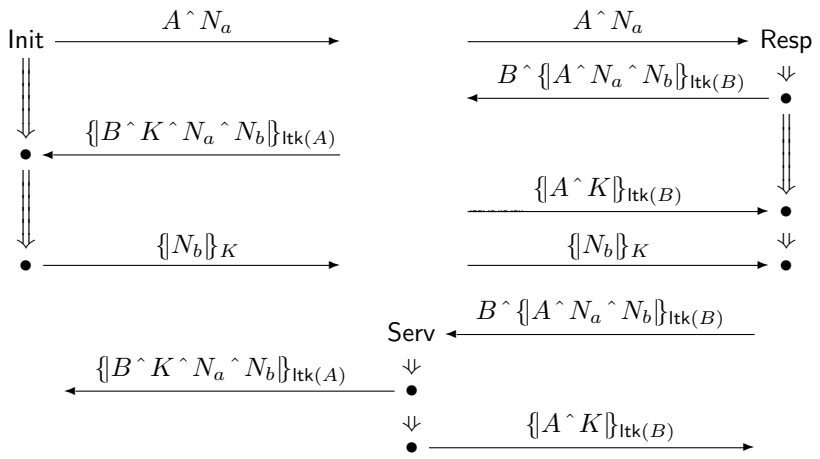

FIGURE 2. Yahalom protocol (forwarding removed)

This means that if we factor H into $\mathbb{A}_0 \overset{H_0}{\mapsto} \mathbb{A}' \overset{H_1}{\mapsto} \mathbb{A}_1$, where $\mathbb{A}'$ is realized, then $\mathbb{A}'$ cannot contain fewer nodes than $\mathbb{A}_1$, or make fewer identifications among atomic values. $\mathbb{A}_1$ is as small and as general as possible. We call $\mathbb{A}_1$ a shape if the homomorphism H is understood (such as an embedding). Shapes exist below realized skeletons: If $H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ with $\mathbb{A}_1$ realized, then the set of shapes H_1 with $H_1 \leq_n H$ is finite and non-empty. (Prop. 5.7.)

The Needham-Schroeder intended run is not a shape for B's strand, because the skeleton shown in Fig. 1 makes fewer identifications among atomic values. Fig. 1 is more general, and the map replacing C with B is nodewise injective.

3. The Yahalom Protocol

The Yahalom protocol [3] (Fig. 2) provides a session key K to principals sharing long-term symmetric keys with a key server. We let $\mathsf{ltk}(\cdot)$ map each principal A to its long term shared key $\mathsf{ltk}(A)$. We assume that all participants agree on the server, which does not also participate as a client.

3.1. **Definition of the Protocol.** The protocol contains three roles, the initiator, the responder, and the server. Each is described by one strand in Fig. 2, and each role is parametrized by A, B, N_a, N_b, K. The parameters are atomic values, and the instances of each role are constructed by replacing them with other atomic values. The behavior Init of the initiator consists in transmitting $A \hat{} N_a$ followed by receiving $\{\!|B \hat{} K \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}$ and finally transmitting $\{\!|N_b|\!\}_K$. The other roles are equally self-explanatory. The key server is trusted to generate a fresh, uniquely originating session key K in each run. By this, we mean that if a skeleton $\mathbb{A}$ contains a server strand with session key K, then $K \in \mathsf{unique}_{\mathbb{A}}$.

We consider the Yahalom protocol in detail, because it requires both of the two types of step in Section 7, and it relies on the outgoing test in the strong form we introduce here. The older form [8] does not suffice. We number the cases we encounter, so the remainder of the paper can refer to them.

3.2. **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. 2. 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. We codify these assumptions in an initial skeleton $\mathbb{A}_0$ consisting of a responder strand, letting $\mathsf{non}_{\mathbb{A}_0} = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$ and $\mathsf{unique}_{\mathbb{A}_0} = \{N_b\}$. What skeletons are shapes for $\mathbb{A}_0$?

We will find that the shape $\mathbb{A}_4$ (Fig. 5) is the only possibility. Any realized $\mathbb{A}$, if it contains a responder strand s, with uncompromised long-term keys and a fresh nonce, must contain an image of $\mathbb{A}_4$. The strand s is in the image of $\mathbb{A}_4$ under a nodewise injective H. So, a portion of $\mathbb{A}$ contains s and resembles $\mathbb{A}_4$.

Transforming the Nonce. B chooses a fresh nonce N_b in its second step (n_0 in Fig. 3), and transmits it within the encrypted unit $\{\!|A \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(B)}$. In B's fourth step (n_2 in Fig. 3), it is received outside that unit, in the form $\{\!|N_b|\!\}_K$. How could it be extracted from the former to appear in the latter? The transformation must begin with case 1 or case 2:

(1) (a) $\{\!|A \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(B)}$ reaches a server strand that transforms it, for some K', into $\{\!|B \hat{} K' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}$. (b) The latter message reaches an initiator strand that transforms it into $\{\!|N_b|\!\}_{K'}$. We ask later whether $K' = K$.

(2) The adversary somehow receives one of the long term shared keys. If the adversary receives $\mathsf{ltk}(B)$, then the original message may never reach the server. If it receives $\mathsf{ltk}(A)$, then the message $\{\!|B \hat{} K' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}$, transmitted by the server, may never reach A.

If the long-term keys do not become available to the adversary, then only the protocol itself can extract N_b from the successive messages, and the protocol does this only by a server strand followed by an initiator strand. Each of cases 1, 2 describes some of the homomorphisms that lead from $\mathbb{A}_0$ to its shapes.

Case 1 describes homomorphisms that add two strands as part of mapping $\mathbb{A}_0$ to a shape $\mathbb{A}'$. In fact, one can factor this mapping into one that maps $\mathbb{A}_0$ into the skeleton $\mathbb{A}_1$ (see Fig. 3) that results from adding these regular strands, and a second homomorphism that does the rest of the work, mapping $\mathbb{A}_1$ to $\mathbb{A}'$. Since K' is the session key originating on a server strand, $K' \in \mathsf{unique}_{\mathbb{A}_1}$.

Turning to Case 2, any homomorphism that does not factor through $\mathbb{A}_1$ depends on a key being compromised. However, $\mathsf{ltk}(A), \mathsf{ltk}(B) \in \mathsf{non}_{\mathbb{A}_0}$, meaning that these keys will be used only on accordance with the protocol, and in particular never transmitted. Thus, they can never be compromised, and Case 2 is vacuous.

Does $K' = K$? The server generated K' and delivered it to A in $\mathbb{A}_1$. B received K, and found it was also used to encrypt the nonce N_b. Must the keys K', K be the same, or could they be distinct?

We again consider transformations of N_b, which has appeared in the forms:

- $\{\!|A \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(B)}$;
- $\{\!|B \hat{} K' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}$, generated from the previous form by the server;
- $\{\!|N_b|\!\}_{K'}$, generated from the previous form by A.

However, we know that N_b also occurs in the form $\{\!|N_b|\!\}_K$ at the node n_1 (as shown in Fig. 3). One of cases 3–5 could account for the transformation:

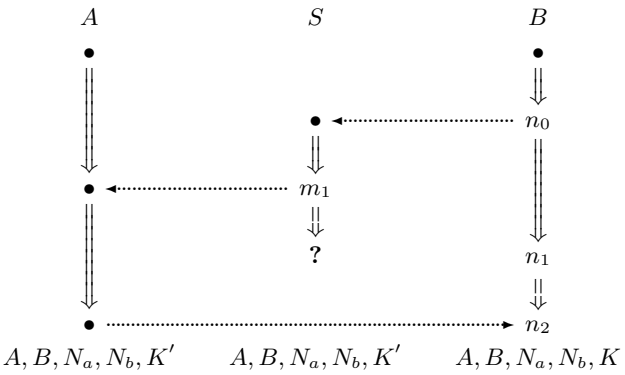

FIGURE 3. Skeleton $\mathbb{A}_1$, with $\mathsf{non}_{\mathbb{A}_1} = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$ and $\mathsf{unique}_{\mathbb{A}_1} = \{N_b, K'\}$.

(3) If $K' = K$, then $\{\!|N_b|\!\}_{K'} = \{\!|N_b|\!\}_K$, and no transformation is needed.
(4) K' could be obtained by the adversary, who decrypts $\{\!|N_b|\!\}_{K'}$, causing N_b to escape from the forms already seen.
(5) Another regular server strand could receive N_b in its original form and retransmit it with a new session key as $\{\!|B \hat{} K'' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}$.

However, we can prune this possibility, because K'' is not usefully different from K'. The messages transmitted by this strand cannot be used in ways different from the messages transmitted by the existing server strand. (Formalized in Proposition 6.11.)

Discarding case 5, we are left with two possibilities: either $K' = K$ or else K' becomes compromised. We exclude case 4 next.

Case 4: K' becomes compromised. Then K' must become available without any protective cryptography. K' originates only at m_1 and is transmitted in that node and possibly also the next node. Thus, it is sent only in the forms $\{\!|B \hat{} K' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}$ and $\{\!|A \hat{} K'|\!\}_{\mathsf{ltk}(B)}$.

If K' occurs without protection, then one of these two cases must hold:

(6) Some instance of one of the $\mathsf{Init}, \mathsf{Resp}, \mathsf{Serv}$ roles can receive K' in one of the forms shown, and retransmit it occurring in some other form.

However, no Yahalom role receives a key and retransmits it in any other form. It is used by the responder as key in preparing an encrypted message, but this cannot lead to its being revealed to the adversary.
(7) One of the keys that protect K' when it is initially transmitted must become compromised. These are the long-term keys $\mathsf{ltk}(A), \mathsf{ltk}(B)$. However, $\mathsf{ltk}(A), \mathsf{ltk}(B) \in \mathsf{non}_{\mathbb{A}_1}$.

So neither case 6 nor case 7 is possible, and K' is uncompromised. Hence $K' = K$.

Let $\mathbb{A}_2$ be the result of replacing K' by K in wherever mentioned in $\mathbb{A}_1$. We now know that if for some homomorphism $H\colon \mathbb{A}_0 \mapsto \mathbb{A}'$ where $\mathbb{A}'$ is a shape, then H is a factors through the homomorphism that embeds $\mathbb{A}_0$ into $\mathbb{A}_2$, which is composed with some other H' that does the rest of the work to get to $\mathbb{A}'$.

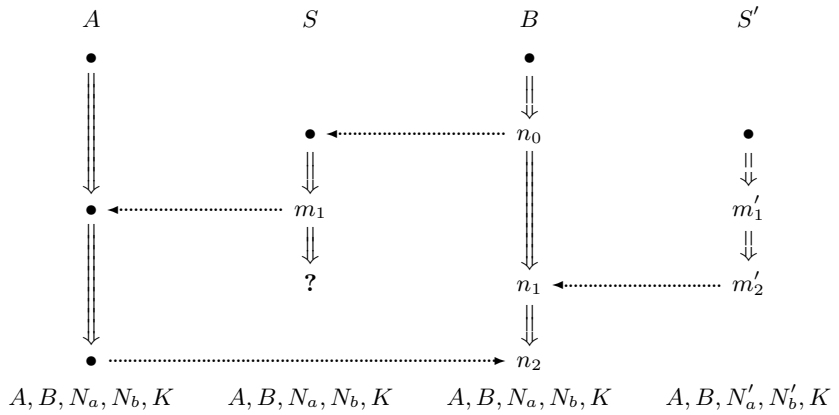

FIGURE 4. $\mathbb{A}_3$, with $\mathsf{non}_{\mathbb{A}_3} = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$ and $\mathsf{unique}_{\mathbb{A}_3} = \{N_b, K\}$.

B's Source for K. The responder B receives $\{\!|A \hat{} K|\!\}_{\mathsf{ltk}(B)}$. How is this term created? There are two possibilities:

(8) A regular strand transmits $\{\!|A \hat{} K|\!\}_{\mathsf{ltk}(B)}$. Only a server strand, with parameters A, B, K, will do so, although we do not know what nonces appear in it. In Fig. 4, we show $\mathbb{A}_3$, which represents what we know in this case.
(9) Alternatively, $\{\!|A \hat{} K|\!\}_{\mathsf{ltk}(B)}$ is generated by the adversary, so $\mathsf{ltk}(B)$ has been compromised. However, $\mathsf{ltk}(B) \in \mathsf{non}_{\mathbb{A}_2}$, excluding this case.

$\mathbb{A}_3$ has an anomaly, however. Although $K \in \mathsf{unique}_{\mathbb{A}_3}$ is intended to originate at just one node, it in fact originates, packaged for A, at both m_1 and m'_1. It is then transmitted later, packaged for B, at m'_2 and possibly also at the location marked **?**, if the latter has happened "already" at the time represented by $\mathbb{A}_3$.

Since $K \in \mathsf{unique}_{\mathbb{A}_3}$, but originates at both m_1 and m'_1, necessarily $m_1 = m'_1$. Hence, the strands marked S and S' are identical. Fig. 5 shows the result of identifying them. As a consequence of identifying these strands, we also infer that $N_a = N'_a$ and $N_b = N'_b$, and that S starts after n_0 and completes before n_1.

Skeleton $\mathbb{A}_4$ is *realized*, since it can really happen with no additional activity of the regular (non-penetrator) participants. Moreover, $\mathbb{A}_4$ is *minimal* in two ways. First, if we leave out any nodes, then either B's strand is no longer embedded in the result, or else the result is no longer realized. Second, we cannot make it more general: If two different strands share a parameter, and we alter that parameter in one of the strands, then the result is no longer realized. For instance, the diagram would no longer fit together if A's parameter N_b were altered to some N'_b. A skeleton that is realized and minimal in these two senses is a *shape*.

Thus, we have shown that $\mathbb{A}_4$ is a shape for $\mathbb{A}_0$. Since all homomorphisms from $\mathbb{A}_0$ to realized skeletons factor through $\mathbb{A}_4$, $\mathbb{A}_4$ is the only shape for $\mathbb{A}_0$.

4. Terms, Strands, and Bundles

In this section and then next (Section 5), we give precise definitions, which include a number of fine points which seemed an unnecessary distraction in the

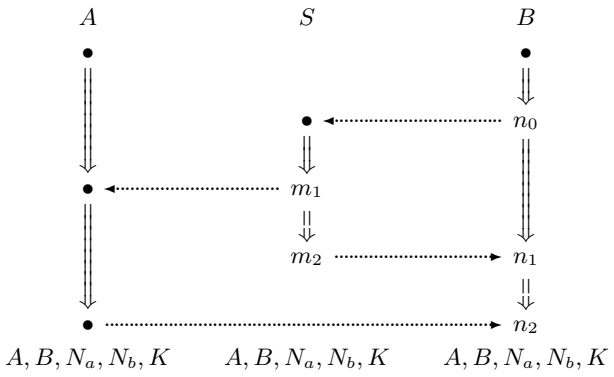

FIGURE 5. Skeleton $\mathbb{A}_4$, with $\mathsf{non}_{\mathbb{A}_4} = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$ and $\mathsf{unique}_{\mathbb{A}_4} = \{N_b, K\}$.

previous sections. Examples of fine points include the tags on concatenated terms in Def. 4.1 and the non-degeneracy conditions involving $\mathcal{O}(s, a)$ in Defs. 4.4 and 5.3.

4.1. **Algebra of Terms.** 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 4.1 (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:*

$$\begin{aligned} (\mathsf{tag}\,\hat{}\,t_0\,\hat{}\,t_1) \cdot \alpha &= \mathsf{tag}\,\hat{}\,(t_0 \cdot \alpha)\,\hat{}\,(t_1 \cdot \alpha) \\ (\{|t|\}_K) \cdot \alpha &= \{|t \cdot \alpha|\}_{K \cdot \alpha} \end{aligned}$$

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* $x \cdot \alpha = x$*.*

4.2. **Strands and Origination.** 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 4.2 (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 4.3. *A* penetrator strand *has trace of one of the following forms:*

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

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

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

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, $t \sqsubseteq \mathsf{msg}(n)$, and $t \not\sqsubseteq \mathsf{msg}(m)$ 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 $\langle -K \rangle$.

4.3. **Protocols and Bundles.**

Definition 4.4 (Protocols)**.** *A* candidate $\langle \Pi, \mathsf{strand\_non}, \mathsf{strand\_unique} \rangle$ *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* $\mathsf{strand\_non}$ *mapping each role* r *to a finite*

set of keys $\mathsf{strand\_non}_r$*, called the non-originating keys of* r*; and (3) a function* $\mathsf{strand\_unique}$ *mapping each role* r *to a finite set of atoms* $\mathsf{strand\_unique}_r$ *called the uniquely originating atoms of* r*.*

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

The regular strands *of* $\langle \Pi, \mathsf{strand\_non}, \mathsf{strand\_unique} \rangle$ *form the set* $\Sigma_\Pi =$

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

The non-originating keys $\mathsf{strand\_non}_r$ and uniquely originating atoms $\mathsf{strand\_unique}_r$ are used in the definition of *augmentations*, Def. 6.3, Clauses 1c,d. The condition that constrains $r \cdot \alpha$ based on $\mathcal{O}(r, a)$ is a non-degeneracy condition. It says that replacement α determines an instance of r only if it does not cause a value a, assumed uniquely originating, to collide with another value already encountered in executing r.

Example 4.5 (Yahalom Protocol)**.** *The Yahalom protocol has a set* Π_Y *of roles containing the three roles shown in Fig. 2 and two listener roles, to hear nonces and keys. For each* $r \in \Pi_Y$*,* $\mathsf{strand\_non}_r = \emptyset$*. For the server role* $\mathsf{Serv} \in \Pi_Y$*,* $\mathsf{strand\_unique}_{\mathsf{Serv}} = \{K\}$*, and for the other roles* $r \in \Pi_Y$*,* $\mathsf{strand\_unique}_r = \mathsf{strand\_non}_r = \emptyset$*.*

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 4.6 (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 $\langle \Pi, \mathsf{strand\_non}, \mathsf{strand\_unique} \rangle$ *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* $a \in \mathsf{strand\_non}_r \cdot \alpha$*, then* a *does not occur in* $\mathcal{B}$*; and (3) if* $s = r \cdot \alpha$ *and* $a \in \mathsf{strand\_unique}_r \cdot \alpha$*, then* a *originates at most once in* $\mathcal{B}$*.*

Example 4.7. *Fig. 1 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 $\mathcal{B}$. Henceforth, assume fixed some arbitrary protocol $\langle \Pi, \mathsf{strand\_non}, \mathsf{strand\_unique} \rangle$.

Proposition 4.8. *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* $b \in \mathsf{strand\_unique}_r \cdot \beta$*, we have* $(\mathcal{O}(s, b)) \cdot \alpha = \mathcal{O}(s \cdot \alpha, b \cdot \alpha)$*. Then* $\mathcal{B} \cdot \alpha$ *is a bundle.*

5. Preskeletons, Skeletons, and Homomorphisms

5.1. **Skeletons.** A preskeleton is potentially the regular (non-penetrator) part of a bundle or of some portion of a bundle. Each of Figs. 1 and 3–5 shows a preskeleton.

A preskeleton consists of nodes annotated with additional information, indicating order relations among the nodes, uniquely originating atoms, and non-originating atoms. We say that an atom a *occurs* in a set nodes of nodes if for some $n \in \mathsf{nodes}$, $a \sqsubseteq \mathsf{msg}(n)$. A key K is *used* in nodes if for some $n \in \mathsf{nodes}$, $\{|t|\}_K \sqsubseteq \mathsf{msg}(n)$. 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 5.1. *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*.*

$\mathbb{A}_3$, shown in Fig. 4, is a preskeleton but not a skeleton, because the session key K originates at two different nodes, lying on the strands labeled S and S'. We converted it into a skeleton by observing that these two strands must in fact be the same.

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, $\mathsf{unique}_{\mathbb{A}} \cap \mathsf{non}_{\mathbb{A}} = \emptyset$.

The skeletons for a particular protocol $\langle \Pi, \mathsf{strand\_non}, \mathsf{strand\_unique} \rangle$ are defined analogously to the bundles for that protocol. A (pre)skeleton $\mathbb{A}$ is a *(pre)skeleton for* protocol $\langle \Pi, \mathsf{strand\_non}, \mathsf{strand\_unique} \rangle$ iff for every $n \in \mathsf{nodes}_{\mathbb{A}}$ with $n = s \downarrow i$, (1) $s \in \Sigma_\Pi$; (2) if $s = r \cdot \alpha$ and $a \in \mathsf{strand\_non}_r \cdot \alpha$, then a does not occur in $\mathbb{A}$; and (3) if $s = r \cdot \alpha$ and $a \in \mathsf{strand\_unique}_r \cdot \alpha$, then $a \in \mathsf{unique}_{\mathbb{A}}$.

5.2. **Skeletons and Bundles.** Bundles correspond to certain skeletons:

Definition 5.2. *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 [8] determine whether $\mathbb{A}$ is realized. Figs. 1 and 5 show realized skeletons, while Fig. 3 shows a non-realized skeleton.

5.3. **Homomorphisms.** When $\mathbb{A}$ is a preskeleton, we may apply a substitution α to it, subject to the same condition as in Prop. 4.8. 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 objects built from it are preskeletons but not skeletons. Preskeleton $\mathbb{A}_3$ in Fig. 4 is built from the skeleton $\mathbb{A}_2$ by adding a strand S' containing an additional point of origination for the session key K; we did this to explain "how B got K." $K \in \mathsf{unique}_{\mathbb{A}_3}$, but K originates at both m_1 and m'_1. In other cases, $\mathbb{A}$ may be a skeleton while $\mathbb{A} \cdot \alpha$ is a preskeleton but not a skeleton. This may occur because $a_1, a_2 \in \mathsf{unique}_{\mathbb{A}}$ have distinct points of origination $n_1, n_2 \in \mathbb{A}$, but $a_1 \cdot \alpha = a_2 \cdot \alpha$. Then the two nodes $n_i \cdot \alpha$ are both points of origination for the common value $a_i \cdot \alpha$.

In a preskeleton, we can sometimes, though, restore the skeleton unique origination property (4′) by a mapping ϕ that carries the two points of origination 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. We applied such a map to $\mathbb{A}_3$ to obtain $\mathbb{A}_4$.

Definition 5.3. *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 $n \in \mathbb{A}_0$, $\mathsf{msg}(\phi(n)) = \mathsf{msg}(n) \cdot \alpha$, with the same direction;*
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 $\phi(\mathcal{O}(s, a)) = \mathcal{O}(s', a \cdot \alpha)$ 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 $\mathbb{A}_0$ to $\mathbb{A}_1$. 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 for $[\phi, \alpha] = [\phi, \alpha']$ implies that the action of α on atoms not mentioned in the $\mathbb{A}_0$ is irrelevant. 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. For instance, a replacement α that sends both N_a and N_b to the same value would not furnish homomorphisms on the examples of Section 2–3. This would amount to the degenerate case in which the responder, expecting to choose a fresh nonce, inadvertently selects the same nonce he has just received.

A homomorphism $I = [\phi, \alpha] \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ is an *isomorphism* iff ϕ is a bijection and α is injective. We say that two homomorphisms H_1, H_2 are isomorphic if they differ by an isomorphism; i.e. $H_1 = I \circ H_2$ for some isomorphism I.

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 $\mathsf{msg}(n), \mathsf{msg}(n')$. For instance, in Section 3.2, when transforming $\mathbb{A}_3$ into $\mathbb{A}_4$, we identified nodes m_1, m_1', and thereby learnt that the additional values N_a', N_b' must respectively equal N_a, N_b. This process could cascade. For instance if N_b' had a point of origination in $\mathbb{A}_4$, then we would have to identify that with the node marked n_1 in Fig. 4. However, when success is possible, and the cascading produces no incompatible constraints, there is a canonical (universal) way to succeed, as follows from the results in Appendix A:

Proposition 5.4. *Suppose $H_0 \colon \mathbb{A} \mapsto \mathbb{A}'$ with $\mathbb{A}$ a preskeleton and $\mathbb{A}'$ a skeleton.*

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 every 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 $\mathbb{A}$, $\mathsf{hull}(\mathbb{A})$.

We say that a skeleton $\mathbb{A}_0$ is *live* if for some $H, \mathbb{A}_1$, $H \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ and $\mathbb{A}_1$ is realized. Otherwise, it is *dead*. There are two basic facts about dead skeletons:

Proposition 5.5 (Dead Skeletons)**.** *(1) If $a \in \mathsf{non}_{\mathbb{A}}$ and $(\mathsf{Lsn}[a]) \downarrow 1 \in \mathbb{A}$, then $\mathbb{A}$ is dead. (2) If $\mathbb{A}$ is dead and $H \colon \mathbb{A} \mapsto \mathbb{A}'$, then $\mathbb{A}'$ is dead.*

5.4. **Shapes.**

Definition 5.6 (Shape)**.** $[\phi, \alpha] \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ *is* nodewise injective *if ϕ is an injective function on the nodes of $\mathbb{A}_0$.*

A homomorphism H_0 is nodewise less than or equal to H_1*, written $H_0 \leq_n H_1$, if for some nodewise injective J, $J \circ H_0 = H_1$. H_0 is* nodewise minimal *in a set of homomorphisms S if $H_0 \in S$ and for all $H_1 \in S$, $H_0 \leq_n H_1$.*

$H \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ is a shape for *$\mathbb{A}_0$ if H is nodewise minimal among the set of homomorphisms $H' \colon \mathbb{A}_0 \mapsto \mathbb{A}_1'$ where $\mathbb{A}_1'$ is realized.*

The composition of two nodewise injective homomorphisms is nodewise injective, and a nodewise injective $H \colon \mathbb{A} \mapsto \mathbb{A}$ is an isomorphism. Thus, H_0, H_1 are isomorphic if each is nodewise less than or equal to the other. Hence, the relation $\leq_n$ is a partial order on homomorphisms to within isomorphism.

If we speak of a skeleton $\mathbb{A}_0$ as nodewise less than another skeleton $\mathbb{A}_1$, we mean that $H \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ for some nodewise injective H. When we say that $\mathbb{A}_1$ is a shape, we mean that it is the target of some shape $H \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$, where a particular $\mathbb{A}_0$ is understood from the context.

Proposition 5.7. *Let $H \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$. The set $\mathcal{S} = \{H' \colon H' \leq_n H\}$ is finite (up to isomorphism). If $\mathbb{A}_1$ is realized, then at least one $H' \in \mathcal{S}$ is a shape for $\mathbb{A}_0$.*

Proof. Letting $H = [\phi, \alpha]$, we generate $\mathcal{S}$ 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.

We associate each location at which a is mentioned with an atom in $\alpha^{-1}(a)$, the inverse image of a under α. An association is permissible if locations containing the same atom in $\mathbb{A}_0$ are associated with the same atom.

The set $\mathcal{S}$ contains the homomorphisms we get given a choice of nodes to omit and a permissible association. H differs by a renaming from a member of $\mathcal{S}$, namely the one that omits no nodes and associates every occurrence of any a with a single representative from $\alpha^{-1}(a)$. Thus, if $\mathbb{A}_1$ is realized, $\mathcal{S}$ has members with a realized target. Letting $\mathcal{S}' \subseteq \mathcal{S}$ be the set of $H' \in \mathcal{S}$ such that the target of H' is realized, $\mathcal{S}'$ is non-empty and finite; hence, $\mathcal{S}'$ has $\leq_n$-minimal 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 Fig. 1.

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. 5.7 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.

6. The Authentication Tests

To direct the process of searching for realized skeletons, we use the *authentication tests* [8] 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) $t_0 \not\sqsubseteq t$; 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) $t = \mathsf{tag}\,\hat{}\,t_1\,\hat{}\,t_2$ 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 $t_0 \sqsubseteq t$ and there is a path from the root to an occurrence of t_0 as a subterm of t that traverses no $t_1 \in S$.

6.1. The Tests in Bundles.

We say that a is *protected* in $\mathcal{B}$ iff $\mathsf{msg}(n) \neq a$ 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 $\mathsf{msg}(n) = a$ 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 4.3), 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 6.1 (Outgoing Authentication Test)**.** *Suppose that* $n_0, n_1 \in \mathcal{B}$*, and*

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

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

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 $\mathsf{msg}(s \downarrow k)$. Moreover, there is a node $m_0 = s \downarrow j$ with $j < i$ such that $a \sqsubseteq \mathsf{msg}(s \downarrow j)$, and $n_0 \preceq_{\mathcal{B}} m_0 \Rightarrow^+ m_1 \preceq_{\mathcal{B}} n_1$.

Proof. Apply Prop. 4.8 to

$$T = \{m \colon m \preceq_{\mathcal{B}} n_1 \text{ and } a \text{ occurs outside } S \text{ in } \mathsf{msg}(m)\}.$$

$n_1 \in T$, 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 $\mathsf{Prot}_{n_1}(\mathcal{B})$, we consider the set $\mathsf{used}(S)$ of keys used for some outermost encryption in S as an approximation, and we speak of an *outgoing test pair for a, S*.

Proposition 6.2 (Incoming Authentication Test)**.** *Suppose that $n_1 \in \mathcal{B}$ is negative, $t = \{\!|t_0|\!\}_K \sqsubseteq \mathsf{msg}(n_1)$, 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. 4.8 to $T = \{m \colon m \preceq_{\mathcal{B}} n_1 \text{ and } t \sqsubseteq \mathsf{msg}(m)\}$. A minimal member $m_1 \in T$ does not lie on a penetrator E-strand because $K \in \mathsf{Prot}(\mathcal{B})$. □

We call m_1 as an *incoming transforming node*, and n_1 an *incoming test node*.

6.2. **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 $H \colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ 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 $\mathbb{A}_1$. This definition uses the protocol origination data n_r, u_r from Def. 4.4.

Definition 6.3 (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 (i) $\preceq_{\mathbb{A}_0}$; (ii) the strand ordering of s up to i; and (iii) pairs (n, m) or (n, m) with $n \in \mathsf{nodes}_{\mathbb{A}_0}$, $m = s \downarrow j$, and $j \leq i$.*

(c) $\mathsf{non}_{\mathbb{A}_1} = \mathsf{non}_{\mathbb{A}_0} \cup (n_r \cdot \alpha)$*; and*

(d) $\mathsf{unique}_{\mathbb{A}_1} = \mathsf{unique}_{\mathbb{A}_0} \cup (u_r \cdot \alpha)$.

(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 1b(iii)) 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. 6.2 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.*

(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* $\mathsf{hull}(\mathbb{A} \cdot \alpha)$*, when the latter is defined. (See Prop. 5.4.)*

We can now state the search-oriented version of Prop. 6.1. 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 $\mathbb{A}_1$.

Since we consider realized skeletons that differ only in their listener strands, we recall that $\mathbb{A}_1 \sim_{\mathsf{L}} \mathbb{A}_2$ if they are both realized and differ only in what listener strands they contain. We will also write $H_1 \sim_{\mathsf{L}} H_2$ if adding listener strands can equalize them; i.e., when the H_i (for $i = 1, 2$) are of the form $H_i\colon \mathbb{A} \mapsto \mathbb{A}_i$, and there are embeddings $E_i\colon \mathbb{A}_i \mapsto \mathbb{A}'$ such that $\mathbb{A}_1 \sim_{\mathsf{L}} \mathbb{A}' \sim_{\mathsf{L}} \mathbb{A}_2$ and $E_1 \circ H_1 = E_2 \circ H_2$.

Theorem 6.4 (Outgoing Augmentation)**.** *Let* $H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$*, where* $\mathbb{A}_1$ *is realized. Let* $n_0, n_1 \in \mathbb{A}_0$ *be an outgoing test pair for* a, S*, 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*;*

(3) *There is a listener augmentation* $H'\colon \mathbb{A}_0 \mapsto \mathbb{A}'_0$ *for some* $K \in \mathsf{used}(S)$*, and a homomorphism* $H''\colon \mathbb{A}'_0 \mapsto \mathbb{A}'_1$ *such that* $H \sim_{\mathsf{L}} H'' \circ H'$*.*

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. 6.1 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 6.4, we prefer to apply Clauses 2, 3 if possible; unnecessary contractions must simply be un-contracted using Prop. 5.7. 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 a fine point missing in the analysis in Section 3. Theorem 6.4 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 use of it in Steps 1(a) and 1(b) 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 8, we may introduce the strands in reverse order, and establish that the server behavior causally preceded the initiator's second action.

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

Theorem 6.5 (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 $\{|t_0|\}_K \sqsubseteq \mathsf{msg}(n_1)$. If $\{|t_0|\}_K$ originates nowhere in $\mathbb{A}_0$, then either:*

1. *$H = H'' \circ \mathsf{hull}_\alpha(\mathbb{A}_0)$ for some contraction α;*
2. *$H = H'' \circ H'$, where H' is an incoming augmentation originating $\{|t_0|\}_K$; or*
3. *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 $a \sqsubseteq \mathsf{msg}(m)$, 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.*

6.3. **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 6.6 (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 $\mathsf{msg}(n_2) \in S$ 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 $n \in \mathcal{N}_G$, $\mathsf{msg}(n) = t$.*

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

Definition 6.7. *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 6.8. *A skeleton $\mathbb{A}$ is realized iff, for every negative $n \in \mathbb{A}$, $\mathsf{msg}(n)$ is penetrator-derivable before n in $\mathbb{A}$.*

Proposition 6.9. *Suppose that $\preceq_{\mathbb{A}}$ reflects origination. If $\mathsf{msg}(n)$ is not penetrator-derivable before n in $\mathbb{A}$, then either:*

(1) *n is an incoming transformed node, i.e., for some $\{|t|\}_K \sqsubseteq \mathsf{msg}(n)$, $K \in \mathsf{non}_{\mathbb{A}} \cup \mathsf{unique}_{\mathbb{A}}$ and K is not penetrator-derivable before n in $\mathbb{A}$; or else*
(2) *(m, n) is an outgoing transformed pair with respect to a, S for (i) some $m \preceq_{\mathbb{A}} n$; (ii) some $a \in \mathsf{unique}_{\mathbb{A}}$ originating at m; (iii) some set S of encrypted terms such that a occurs only within S in $\mathsf{support}(n)$; and (iv) for each $K \in \mathsf{used}(S)$, K^{-1} is not penetrator-derivable before n in $\mathbb{A}$.*

Proof. Similar to[8, Prop. 7]. □

Recall that shapes (being minimal) do not contain listener strands, so Clause 3 of Theorems 6.4, 6.5 need not appear in the following:

Theorem 6.10 (Authentication Tests Completeness)**.** *Let $J = [\phi, \alpha] \colon \mathbb{A} \mapsto \mathbb{A}_s$ be a shape. J is isomorphic to $H_i \circ \ldots \circ H_0$ for some sequence of homomorphisms $\{H_j\}_{0 \le j \le i}$, where*

(1) *$H_0 \colon \mathbb{A} \mapsto \mathbb{A}_0$ is surjective and $\mathbb{A}_0$ is a substructure of $\mathbb{A}$, or a contraction of a substructure of $\mathbb{A}$; and*
(2) *For each j with $1 \le j \le i$, $H_j \colon \mathbb{A}_{j-1} \mapsto \mathbb{A}_j$ is a contraction or an augmentation as in Theorem 6.4 or Theorem 6.5, Clauses 1, 2.*

Proof. We define two sequences of homomorphisms, namely $\{H_j\}_{0 \le j \le i}$ and $\{L_j\}_{0 \le j \le i}$, such that (1), (2) hold, and moreover, (3) $J = L_j \circ H_j \circ \ldots \circ H_0$, and (4) each L_j is nodewise injective and L_i is an isomorphism. (3) and (4) imply that J is isomorphic to the composition of the H_j.

By the definition of shape, if any composition $H_j \circ \ldots \circ H_0$ is realized, then we may take $j = i$ and stop. The nodewise injective L_j must be an isomorphism.

First, we define H_0 to prune unnecessary strands in $\mathbb{A}$, so that L_0 will be node injective. Partition the strands in $\mathbb{A}$ by their image under ϕ; i.e. $e_s = \{s' \colon \phi(s' \downarrow 1) = \phi(s \downarrow 1)\}$. For each partition element e_s, choose a representative $r(e_s)$ of maximal height. We know that α unifies all the terms on the strands in any partition element, so there is a most general contraction β compatible with these identifications. Enrich the ordering to reflect origination. Let $H_0 = [(\lambda s \,.\, r(e_s)), \beta]$.

Next, suppose that $H_0 \ldots H_j$ and $L_0 \ldots L_j$ have been defined, with $H_j \colon \mathbb{A}_{j-1} \mapsto \mathbb{A}_j$, and $\mathbb{A}_j$ is not realized. Let $L_j = [\phi_j, \beta_j] \colon \mathbb{A}_j \mapsto \mathbb{A}_s$. Let $n_1 \in \mathbb{A}_j$ be a negative node with $\mathsf{msg}(n_1)$ not penetrator derivable before n_1 in $\mathbb{A}_j$ (Prop. 6.8).

By Prop. 6.9, n_1 is either an unsolved incoming transformed node for some $\{|t|\}_K$ or else half of an unsolved outgoing transformed pair (n_0, n_1). In the latter case, we choose n_0 to be the point of origination of some $a \in \mathsf{unique}_{\mathbb{A}_j}$ such that $a \sqsubseteq \mathsf{msg}(n_1)$, and (n_0, n_1) is an outgoing transformed edge for a, S for some S. There are now the following possibilities.

(1) $\phi_j(n_1)$ is still unsolved, meaning that $K \cdot \beta_j$ (or some $K = K_1 \cdot \beta_j$ for some $K_1^{-1} \in \mathsf{used}(S)$) is compromised. Since n_1 is not penetrator-derivable in $\mathbb{A}_j$, some such K is not derivable.

If for some S_1, $\mathbb{A}_s \setminus J_j(\mathbb{A}_j)$ contains an outgoing transforming edge for $K \cdot \beta_j, S_1 \cdot \beta_j$, then we augment $\mathbb{A}_j$ with a most general preimage of this edge. If required, first apply a contraction H_{j+1} to $\mathbb{A}_j$. Then the augmentation is H_{j+2}, and J_{j+2} is J_{j+1} together with this addition. Otherwise, the augmentation is H_{j+1}.

If $\mathbb{A}_s$ contains no additional outgoing transforming edge for $K \cdot \beta_j$, then this K is already derivable in $\mathbb{A}_j$, contradicting the choice of K.

(2) Outgoing transformed edge $\phi_j(n_0), \phi_j(n_1)$:
   (a) $\phi_j(n_0), \phi_j(n_1)$ is no longer a transformed edge with respect to $a \cdot \beta_j, S \cdot \beta_j$. In this case, let H_{j+1} be a most general contraction with this property.
   (b) $\phi_j(n_0), \phi_j(n_1)$ is solved in $\mathbb{A}_s$. Select a transforming edge contained in $\mathbb{A}_s$. Let $m_0 \Rightarrow^+ m_1$ be a most general preimage of the outgoing transforming edge. If $m_0 \Rightarrow^+ m_1$ is not a transforming edge for (n_0, n_1) and a, S, then the reason is that $m_0 \Rightarrow^+ m_1$ is less general than (n_0, n_1). In this case, first contract $\mathbb{A}_j$. After contracting, one will at the next step augment with $m_0 \Rightarrow^+ m_1$. If contraction is not needed, augment $\mathbb{A}_j$ with $m_0 \Rightarrow^+ m_1$.

(3) $\phi_j(n_1)$ is a solved incoming test node in $\mathbb{A}_s$. Select a transforming node contained in $\mathbb{A}_s$. Let m_1 be a most general preimage of the incoming transforming node. If m_1 is not a transforming node for n_1, then the reason is that the transforming node m_1 is not as general as n_1. In this case, first contract $\mathbb{A}_j$. After contracting, one will at the next step augment with m_1. If contraction is not needed, augment $\mathbb{A}_j$ with m_1.

Thus, if $\mathbb{A}_j$ is realized, it is isomorphic to $\mathbb{A}_s$; otherwise, we extend $\{H_j\}, \{L_j\}$. □

6.4. **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) $\mathsf{nodes}_{\mathbb{A}'_0} \setminus \mathsf{nodes}_{\mathbb{A}_0} = \{s' \downarrow j \colon j \leq i\}$ 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 6.11. *Suppose* $\mathbb{A}'$ *augments* $\mathbb{A}$ *with a copy of* s*, namely* s'*. Let* $J' = [\phi', \alpha'] \colon \mathbb{A}' \mapsto \mathbb{A}'_s$ *with* $\mathbb{A}'_s$ *a shape. Then* $\phi'(s' \downarrow j) = \phi'(s \downarrow j)$.

Proof. Let $H'_i \circ \ldots \circ H'_0$ be the decomposition of J'. We may now construct a corresponding sequence $H_i \circ \ldots \circ H_0$ starting from $\mathbb{A}$, but using the identity in place of any steps H'_j such that the non-derivable node is not present in $\mathbb{A}_j$. For any node whose derivation uses positive nodes from s', we use the corresponding positive nodes in s. Thus, the target $\mathbb{A}_s$ of $H_i \circ \ldots \circ H_0$ is realized, and a substructure of $\mathbb{A}'_s$. By the definition, $H_i \circ \ldots \circ H_0 \circ I_0$ is a homomorphism from $\mathbb{A}'$ to $\mathbb{A}_s$. Since $\mathbb{A}_s$ is embedded in $\mathbb{A}'_s$, there's a node injective map from $\mathbb{A}_s$ to $\mathbb{A}'_s$. If this is not the identity, it contradicts $\mathbb{A}'_s$ being a shape. □

7. Search

In this section, we describe how to search for shapes, abstracting the individual types of steps in Section 7.1, and describing the control strategy in Section 7.2.

7.1. **Search Steps.** There are two types of search steps, *outgoing steps* and *incoming steps.*

Outgoing. The outgoing step states that each outgoing test pair n_0, n_1 must be solved, either by contracting atoms, or by adding an outgoing transforming edge or a listener strand.

We contracted atoms, thus eliminating the outgoing test pair, when letting $K' = K$ in case 3. We added outgoing transforming edges, namely the server and initiator strands that transform the nonce, twice in cases 1(a) and 1(b). We added listener strands repeatedly when checking for compromise.

The check for K' added another outgoing test pair between node m_1 in Fig. 3 and the listener strand receiving K'. This outgoing pair was unsolvable (case 6), showing K' uncompromised.

Outgoing test principle. *Let $H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ with $\mathbb{A}_1$ realized, and let $n_0, n_1 \in \mathbb{A}_0$ be an outgoing test pair for a and S. If $\mathbb{A}_0$ contains (preceding n_1) no outgoing transforming edge for a, S, then, for some H'', $H = H'' \circ H'$ where either:*

1. *H' is a contraction; or*
2. *$H'\colon \mathbb{A}_0 \mapsto \mathbb{A}'$ is an embedding adding $m_0 \Rightarrow^+ m_1$, an outgoing transforming edge for a, S, where $n_0 \preceq_{\mathbb{A}'} m_0$ and $m_1 \preceq_{\mathbb{A}'} n_1$; or*
3. *H' is an embedding adding $\mathsf{Lsn}[K^{-1}]$, for some $K \in \mathsf{used}(S)$.*

Clause 1 is used when $(H(n_0), H(n_1))$ is no longer an outgoing test pair, as when we contracted K' to K in Yahalom, case 3. It is also sometimes needed to prepare for an application of Clause 2, if (n_0, n_1) is more general than some transforming edge in a protocol role. Then the contraction H *unifies* a member of S with a subterm of a role node. Clause 1 is needed only in these two cases. Clause 3 uses the inverse K^{-1} because in public-key (asymmetric) algorithms, we regard $\mathsf{pubk}(A), \mathsf{privk}(A)$ as inverses; symmetric keys are self-inverse.

Older forms of the outgoing test [8] did not contain the set parameter S. They applied only to singleton S. Yahalom case 1(b) requires a non-singleton S, for instance.

There are only finitely many homomorphisms (to within isomorphism) that satisfy any of Clauses 1–3, because only finitely many atoms can be mentioned in $\mathbb{A}_0$ and only finitely many transforming edges are available in one protocol. In particular, there is a finite set of most general homomorphisms for a given test. A set of homomorphisms $\{H_k\}_{k \le j}$ is an *outgoing cohort* if for every H' satisfying Clauses 1–3, there is some $k \le j$ and some H'' such that $H' = H'' \circ H_k$

Cases 1(a)–2 form an outgoing cohort. So do cases 1(b)–2, 3–5, and 6–7.

Incoming. The incoming step states that when an encryption $\{|t|\}_K$ is received, then either some regular strand is responsible, or else K is compromised. An incoming step was used in Section 3.2 to provide B's source for K.

Incoming test principle. *Let $H\colon \mathbb{A}_0 \mapsto \mathbb{A}_1$ with $\mathbb{A}_1$ realized, and let $n_1 \in \mathbb{A}_0$ receive message $\{|t|\}_K$. If $\mathbb{A}_0$ contains (preceding n_1) no m_1 transmitting $\{|t|\}_K$, then, for some H'', $H = H'' \circ H'$ where either:*

(1) H' *is a contraction; or*
(2) $H' \colon \mathbb{A}_0 \mapsto \mathbb{A}'$ *is an embedding adding an* $m_1 \preceq_{\mathbb{A}'} n_1$ *transmitting* $\{|t|\}_K$ *; or*
(3) H' *is an embedding adding* $\mathsf{Lsn}[K]$.

Here one uses Clause 1 only to prepare for an application of Clause 2, if n_1 is more general than a node in a role of the protocol. Then the contraction H *unifies* $\{|t|\}_K$ with a subterm of a role node.

Again, there are finite sets $\{H_k\}_{k \leq j}$ such that $H' = H'' \circ H_k$ for every H' satisfying Clauses 1–3, some $k \leq j$, and some H''; we call them *incoming cohorts.*

Cases 8–9 form an incoming cohort. Given an outgoing or incoming cohort $\{H_k\}_{k \leq j} = H_k \colon \mathbb{A} \mapsto \mathbb{A}_k$, we call the skeletons $\{\mathbb{A}_k\}_{k \leq j}$ a cohort also.

We can now state a third important fact about dead skeletons, a consequence of Prop 5.5 Clause 5.5.

Proposition 7.1. *If* $\mathbb{A}$ *is not realized, and, moreover, all the* $\mathbb{A}'_k$ *in an incoming or outgoing cohort are dead, then* $\mathbb{A}$ *is dead.*

7.2. **Search Strategy.** The goal of CPSA is defined using in terms of a binary relation, a unary predicate, and a function:

$\mathsf{step}(\mathbb{A}, C)$**:** holds if C is a finite set of skeletons forming an outgoing or incoming cohort for $\mathbb{A}$. Any homomorphism from $\mathbb{A}$ to a realized skeleton passes through some $\mathbb{A}_k \in C$. The principles of Section 7.1 imply that the tests and their cohorts may be used in any order, while still finding all shapes.

$\mathsf{realized}(\mathbb{A})$**:** holds if $\mathbb{A}$ is realized, which we can determine directly.

$\mathsf{min\_real}_{\mathbb{A}_0}(\mathbb{A}')$**:** is defined if $\mathbb{A}'$ is realized. Its value is the finite set of skeletons $\mathbb{A}$ such that (1) there is a homomorphism from $\mathbb{A}_0$ to $\mathbb{A}$; (2) $\mathbb{A}$ is realized; (3) there is a nodewise injective homomorphism from $\mathbb{A}$ to $\mathbb{A}'$; and (4) $\mathbb{A}$ is $\leq_n$-minimal among skeletons satisfying (1–3). Prop. 5.7 shows that this set is finite and the proof indicates how to find its members.

We say $\mathsf{child}(\mathbb{A}, \mathbb{A}')$ if for some C, $\mathsf{step}(\mathbb{A}, C)$ and $\mathbb{A}' \in C$. Let $\mathsf{descendent}$ be the reflexive, transitive closure of child. The goal of the search, given a starting skeleton $\mathbb{A}_0$, is to determine the set

$$\mathsf{shapes}(\mathbb{A}_0) = \{\mathbb{A}_2 \colon \exists \mathbb{A}_1 \,.\, \mathsf{descendent}(\mathbb{A}_0, \mathbb{A}_1) \ \wedge \ \mathbb{A}_2 \in \mathsf{min\_real}_{\mathbb{A}_0}(\mathbb{A}_1)\}.$$

To do so, we use the search algorithm in Fig 6. We also need some auxiliaries:

$\mathsf{dead}(\mathbb{A})$**:** holds if $\mathbb{A}$ cannot ever be realized, i.e. there is no $H \colon \mathbb{A} \mapsto \mathbb{A}'$ for any realized $\mathbb{A}'$. $\mathsf{Dead}(\mathbb{A})$ follows from any of the following: (1) $\mathbb{A}$ contains $\mathsf{Lsn}[a]$ where $a \in \mathsf{non}_{\mathbb{A}}$; (2) $\mathsf{dead}(\mathbb{A}_0)$ and $H \colon \mathbb{A}_0 \mapsto \mathbb{A}$; or (3) $\mathsf{step}(\mathbb{A}, C)$ where C consists of dead skeletons. (See Props. 5.5, 7.1.)

$\mathsf{redundant\_strand}(\mathbb{A})$**:** tests whether $\mathbb{A}$ contains a redundant strand that can be identified with some other strand by a homomorphism from $\mathbb{A}$ to a proper subskeleton. Prop. 6.11 justifies discarding $\mathbb{A}$, as we did in Yahalom, case 5.

$\mathsf{step\_applies}(\mathbb{A})$**:** tests if an unsolved outgoing or incoming step exists in $\mathbb{A}$.

$\mathsf{apply\_step}(\mathbb{A})$**:** selects an unsolved step, finds an incoming or outgoing cohort, updates the step relation, and then returns the cohort, assuming $\mathsf{step\_applies}(\mathbb{A})$ is true.

$\mathsf{targets}(\vec{H})$**:** $= \{\mathbb{A}_k \colon k \leq j\}$, if $\vec{H}$ is a set of j homomorphisms $H_k \colon \mathbb{A} \mapsto \mathbb{A}_k$.

We assume $\mathsf{select}\ \mathcal{S}$ selects a member of $\mathcal{S}$ if it is non-empty; and $\mathsf{filter}\ p\ \mathcal{S}$ takes the subset of $\mathcal{S}$ satisfying p. The failure marked "Impossible" in Fig. 6 cannot be

```
F := {A_0};   shapes := ∅;   seen := F;
while F ≠ ∅ begin
      A := select(F);   F := F \ {A};
      if realized(A)
        then shapes := shapes ∪ min_real_{A_0}(A)
      else if redundant_strand(A) then continue
      else if step_applies(A)
           then begin
                let new = targets(apply_step(A)) \ seen in
                F := F ∪ new;
                F := F \ (filter dead F);
                seen := seen ∪ new
           end
      else fail "Impossible."
      end;
return shapes
```

FIGURE 6. CPSA Search Algorithm

reached, because the completeness result (Prop. 6.9) ensures that when $\mathbb{A}$ is not realized, then some authentication test step applies.

8. Implementing CPSA

We discuss here aspects of the CPSA implementation that seem interesting. They are: finding candidate transformations in protocols, and using unification in applying them; choosing sets S for outgoing tests, and representing the sets; and a few items for future work.

Finding transformations. When CPSA reads a protocol description from a textual source, it identifies all the potential transforming edges. For the outgoing tests, it locates all candidate pairs of a reception node m_0 and a transmission node m_1 later on the same role such that a key or nonce is received in one or more encrypted forms on m_0 and retransmitted outside these forms in m_1. For incoming tests, CPSA notes all transmission nodes m_1 that send encrypted units.

To find outgoing transforming edges for $a \in \mathsf{unique}_{\mathbb{A}}$ and a set S, CPSA considers each candidate pair (m_0, m_1). Suppose an encrypted sub-message t of $\mathsf{msg}(m_0)$ unifies with a member of S using a replacement α. If $a \cdot \alpha$ occurs in $\mathsf{msg}(m_0) \cdot \alpha$, but only within $S \cdot \alpha$, then we check $\mathsf{msg}(m_1) \cdot \alpha$. If it occurs outside $S \cdot \alpha$ in $\mathsf{msg}(m_1) \cdot \alpha$, then (m_0, m_1) is a successful candidate. If α identifies atoms, so that it contracts S, then we apply the Outgoing Test Principle twice, once to apply this contraction, and once to add the instance of $m_0 \Rightarrow^+ m_1$. We also check whether a contraction eliminates the outgoing test edge entirely.

For incoming tests, we do a unification on the candidate nodes m_1.

Selecting sets S for outgoing tests. To select sets S in the Outgoing Test Principle, we settled on a trick we call the "forwards-then-backwards" technique. CPSA plans a sequence of applications of applications of it, like Yahalom cases 1(a) and 1(b), until no further transforming edge is found. It follows the transmission of the

Protocol	Point of view	Runtime	Shapes
ISO reject	responder	0.193s	2
Kerberos	client	1.443s	1
Needham-Schroeder	responder	0.055s	1
Needham-Schroeder-Lowe	responder	0.124s	1
Yahalom	responder	2.709s	1

FIGURE 7. Protocols with CPSA runtimes

uniquely originating value—N_b in that case—forwards, treating newly introduced atoms like K' as free variables.

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

CPSA uses the sets in the opposite order, i.e. S_2 is used first to introduce the initiator strand in Fig. 3. Then S_1 is used to interpolate the first two nodes of the server strand between n_0 and the initiator strand. This order respects the precedence relations given in the Outgoing Test Principle.

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. The family of candidate Ss is closed under the operations of union and set difference. The primitive members are either singletons $\{t_0\}$ or generic sets, which represent all the instances of a term t_1 generated when certain of t_1's parameters vary (respecting types). Thus, the family of candidate sets S are represented as finite unions and differences of values of the form $\lambda\vec{v}\,.\,t$, where the vector $\vec{v}$ binds 0 or more atoms in t. Restricting S to the sets representable in this form does not falsify the completeness property. (See the proof of Prop. 6.9.)

This representation fits nicely with our use of unification to provide an extremely focused search, while preserving the completeness property. The focused search allows good runtimes on a variety of protocols. 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. 7. CPSA is implemented in OCaml.

Future work. Future work focuses on expanding beyond the bare-bones Dolev-Yao model described above. We intend to augment CPSA with Diffie-Hellman operations, as studied in [9]. We will also allow keys to be complex messages, typically the result of hashing. In our current framework, replacements map atoms to other atoms only, but it should be possible to map atoms to more general terms, at the cost of using more sophisticated methods to check whether skeletons are realized (e.g. [12]). The skeletons-and-homomorphisms approach may remain useful in a cryptographic, asymptotic probabilistic context.

Acknowledgments. Lenore Zuck and John D. Ramsdell made very valuable comments on earlier drafts. Larry Paulson first pointed out to us how good an analysis challenge the Yahalom protocol is.

REFERENCES

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

[2] 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.
[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] 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.
[6] Andrew D. Gordon and Alan Jeffrey. Types and effects for asymmetric cryptographic protocols. *Journal of Computer Security*, 12(3/4):435–484, 2003.
[7] 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.
[8] Joshua D. Guttman and F. Javier Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 283(2):333–380, June 2002.
[9] 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.
[10] Leslie Lamport. Time, clocks and the ordering of events in a distributed system. *CACM*, 21(7):558–565, 1978.
[11] 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.
[12] 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.
[13] Roger Needham and Michael Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), 1978.
[14] Lawrence C. Paulson. Relations between secrets: Two formal analyses of the Yahalom protocol. *Journal of Computer Security*, 2001. Also available as Cambridge University Computer Laboratory Technical Report 432 (1997).
[15] 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.
[16] 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.

Appendix A. Unifying Equivalence Relations

The essential content of Prop. 5.4 is that there is a coarsest equivalence relation that satisfies the cascading conditions mentioned in Section 5.3.

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 A.1. *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 A.2. *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 $\mathsf{msg}(s \downarrow k)\,\mathbf{R}\,\mathsf{msg}(s' \downarrow k)$.*

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 A.3. *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 A.4. *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 A.5. *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 A.6. *An equivalence relation* $\mathbf{R}$ *on atoms* collapses *a pre-skeleton* $\mathbb{A}$ *iff for every atom* a*, the set of strands*

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

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

Proposition A.7. *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 A.8. *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.*

Contents

1. Introduction 1
2. A Small Example with the Core Ideas 2
2.1. Terminology 2
2.2. The NS Shape 3
2.3. Skeletons, Homomorphisms, Shapes 4
3. The Yahalom Protocol 5
3.1. Definition of the Protocol 5
3.2. Yahalom: Shapes for the Responder 6
4. Terms, Strands, and Bundles 8
4.1. Algebra of Terms 9
4.2. Strands and Origination 10
4.3. Protocols and Bundles 10
5. Preskeletons, Skeletons, and Homomorphisms 11
5.1. Skeletons 11
5.2. Skeletons and Bundles 12
5.3. Homomorphisms 13
5.4. Shapes 14
6. The Authentication Tests 15
6.1. The Tests in Bundles 15
6.2. The Tests in Skeletons 16
6.3. Completeness of the Authentication Tests 18
6.4. A Pruning Condition 20
7. Search 21
7.1. Search Steps 21
7.2. Search Strategy 22
8. Implementing CPSA 23
References 24
Appendix A. Unifying Equivalence Relations 25