Searching for Shapes in Cryptographic Protocols*

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

The MITRE Corporation

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 [7]. 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 [10]. A *skeleton* represents regular (nonpenetrator) 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 [13] or fragments of message sequence charts [12]. A skeleton is *realized* if it is nonfragmentary, 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*

* Supported by the National Security Agency and by MITRE-Sponsored Research.
shaddin@stanford.edu, {guttman, jt}@mitre.org. TACAS (LNCS), March 2007.

for shapes using the authentication tests [10] 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. It cannot terminate for every protocol [8]. It does, however, terminate for reasonably inclusive classes [4, 19].

The type-and-effect system for spi calculus [9] is 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. Blanchet's ProVerif [1] is also based on a sound approximation, and may thus refuse to certify a protocol even though there are no counterexamples.

CPSA's search is related to the second version of Athena [18], which adopted the authentication tests from [10]. However, CPSA differs from Athena in several ways. First, it involves the regular behaviors alone; we never represent adversary activity within a shape. Second, the notion of shape defines a criterion for which possible executions should be considered, among the infinitely many executions (of unbounded size) of any protocol. Third, we introduce strong versions of the authentication tests, for which completeness is true.

The shapes describe protocol executions of all sizes; we do not follow the widely practiced *bounded* protocol analysis (e.g. [2, 15]).

Structure of this paper. We develop the CPSA search strategy from examples, leaving precise definitions, theorems, and proofs to an extended version [6]. Section 2 shows a protocol and its shapes, and introduces terminology. Section 3 introduces the Yahalom protocol [17], a more substantial example. The search for shapes is guided by the authentication test principles (Section 4), which we apply to analyze Yahalom's protocol in Section 5. This analysis illustrates almost every aspect of the CPSA search method. In Section 6, we define the search's control structure. The CPSA implementation is the subject of Section 7.

2 Shapes: the Core Idea

In practice, protocols have remarkably few shapes. The Needham-Schoeder-Lowe [16, 14] protocol has only one. A responder B, asking what global behavior must have occurred when B has had a local run of the protocol, finds the initiator A must have had a matching run. An initiator A knows that B must have had a matching run, although the last message may not have been received.

Uniqueness of shape is unsurprising for so strong a protocol. However, even a flawed protocol such as the original Needham-Schroeder may have a unique shape, shown in Fig. 1.

Terminology. Newly introduced terminology is in **boldface**.

B's local behavior is represented by the right-hand column in Fig. 1, consisting of nodes connected by double arrows $\bullet \Rightarrow \bullet$. A's local behavior is represented

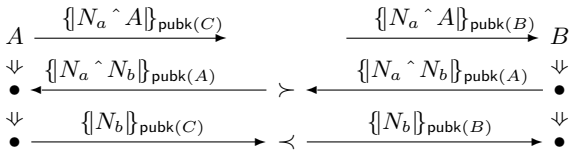

Fig. 1. Needham-Schroeder Shape for B ($\mathsf{privk}(A)$ uncompromised, N_b fresh)

by the left-hand column. We call such a column a **strand**. The **nodes** represent 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. A *local behavior* as used so far refers to a regular strand.

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

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. **Penetrator strands** codify the basic abilities that make up the Dolev-Yao model. They include transmitting an atomic value such as a nonce or a key; transmitting an encrypted message after receiving its plaintext and key; and transmitting a plaintext after receiving ciphertext and decryption key. The adversary can 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. The penetrator strands are independent of the protocol under analysis.

Let $\mathcal{B}$ be a finite, directed acyclic graph whose nodes lie on regular and penetrator strands, and whose edges are either (a) strand succession edges $n_0 \Rightarrow n_1$, or else (b) message transmission edges $n \rightarrow m$ where $\mathsf{msg}(n) = \mathsf{msg}(m)$, n is a transmission node, and m is a reception node. $\mathcal{B}$ is a **bundle** if (1) if $n_0 \Rightarrow n_1$ and $n_1 \in \mathcal{B}$, then $n_0 \in \mathcal{B}$, and (2) for every reception node $m \in \mathcal{B}$, there is a unique transmission node $n \in \mathcal{B}$ such that the edge $n \rightarrow m$ is in $\mathcal{B}$. The conditions (1,2) ensure that $\mathcal{B}$ is causally well founded. A *global behavior* or *execution*, as used so far, refers to a bundle.

The NS Shape. Suppose B's nonce N_b has been freshly chosen, and A's private key $\mathsf{privk}(A)$ is uncompromised. In this protocol, $\mathsf{privk}(A), \mathsf{privk}(B)$ are used only to destructure incoming messages, never to construct messages. Given that—on a particular occasion—B received and sent the messages in the strand shown at the right in Fig. 1, what must have occurred elsewhere in the network?

A must have had a partially matching strand, with the messages sent and received in the order indicated by the arrows of both kinds and the connecting symbols $\prec$. These symbols mean that the endpoints are ordered, but that other behavior may intervene, whether adversary strands or regular strands. A's strand is only partially matching, because the principal A meant to contact is some C which may or may not equal B. There is no alternative: Any diagram containing the responder strand of Fig. 1 must contain at least an instance of 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 some bundle, forming a *minimal* set containing initial regular strands (in this case, the right-hand column). Possible bundles 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. Nothing useful would follow without any such assumptions.

Although there is a single shape, there are two ways that this shape may be realized in bundles. 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 [14]; 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.

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 bundle $\mathcal{B}$. Every message received by a regular participant either should have been sent previously, or should be constructable by the adversary using messages sent previously. Fig. 1 shows a skeleton $\mathbb{A}_{ns}$, indeed a realized one.

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 atoms to atoms. We write $t \cdot \alpha$ for the result of applying a replacement α to all the atoms mentioned in a message t. $H = (\phi, \alpha)$ is a homomorphism iff: (1) ϕ respects strand structure, and for all $n \in \mathbb{A}_0$, $\mathsf{msg}(n) \cdot \alpha = \mathsf{msg}(\phi(n))$; (2) $m \preceq_{\mathbb{A}_0} n$ implies $\phi(m) \preceq_{\mathbb{A}_1} \phi(n)$; (3) $\mathsf{non}_{\mathbb{A}_0} \cdot \alpha \subseteq \mathsf{non}_{\mathbb{A}_1}$; and (4) $\mathsf{unique}_{\mathbb{A}_0} \cdot \alpha \subseteq \mathsf{unique}_{\mathbb{A}_1}$.

Homomorphisms are *information-preserving* transformations. Each skeleton $\mathbb{A}_0$ describes the realized skeletons reachable from $\mathbb{A}_0$ by homomorphisms. Since

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

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, consider the map H_{ns} embedding a single strand of Fig. 1 (e.g. $\mathbb{A}_b$ containing only B's strand on the right side) into $\mathbb{A}_{ns}$. This is a homomorphism $H_{ns}\colon \mathbb{A}_b \mapsto \mathbb{A}_{ns}$. Likewise if we embed the first two nodes of B's strand (rather than all of $\mathbb{A}_b$) into $\mathbb{A}_{ns}$. Another homomorphism H_i rewrites each occurrence of C in $\mathbb{A}_{ns}$ to B, hence each occurrence of $\mathsf{pubk}(C)$ to $\mathsf{pubk}(B)$. It yields the Needham-Schroeder intended run $\mathbb{A}_{nsi}$.

A homomorphism $H = (\phi, \alpha)$ is **nodewise injective** if the function ϕ on nodes is injective. The nodewise injective homomorphisms determine a partial order on homomorphisms: If for some nodewise injective H_1, $H_1 \circ H = H'$, 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. If H is a shape, and we can 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 identify fewer atomic values. $\mathbb{A}_1$ is as small and as general as possible.

We call a *skeleton* $\mathbb{A}_1$ a shape when the homomorphism H (usually an embedding) is understood. In this looser sense, Fig. 1 shows the shape $\mathbb{A}_{ns}$. Strictly, the embedding $H_{ns}\colon \mathbb{A}_b \mapsto \mathbb{A}_{ns}$ is the shape. The embedding $H_{nsi}\colon \mathbb{A}_b \mapsto \mathbb{A}_{nsi}$, with target the Needham-Schroeder intended run $\mathbb{A}_{nsi}$, is not a shape. $\mathbb{A}_{ns}$ identifies fewer atoms, and the map replacing C with B is a nodewise injective $H_i\colon \mathbb{A}_{ns} \mapsto \mathbb{A}_{nsi}$, so $H_{ns} \leq_n H_i \circ H_{ns} = H_{nsi}$.

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.

3 The Yahalom Protocol Definition

The Yahalom protocol (Fig. 2 [17]) 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.

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 some message of the form $\{\!|B \hat{} K \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}$ and finally transmitting $\{\!|N_b|\!\}_K$. The other roles are also 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 $K \in \mathsf{unique}_{\mathbb{A}}$.

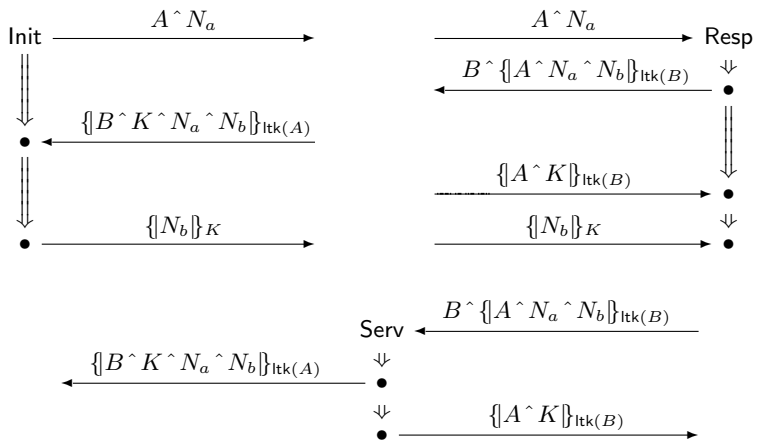

Fig. 2. Yahalom protocol (forwarding removed)

4 Search Steps

The authentication tests are the basic steps leading from a particular initial skeleton to its shapes. The Yahalom protocol requires both types of step, and relies on the strong outgoing test we give here. The older form [10] does not suffice.

Terminology. A **protocol** is a finite set of regular strands, called the **roles**. For instance, the Yahalom roles include the three strands shown in Fig. 2. Roles have atoms as parameters, namely A, B, N_a, N_b, K for each role of Fig. 2. A parameter may be distinguished by the assumption that it is always uniquely originating, like the session key K in the Yahalom server role. The instances of roles under replacements are regular strands.

We assume that each protocol also includes **listener strand** roles, by which we mean a regular strand with a single node receiving an atomic message. We write $\mathsf{Lsn}[a]$ for the strand $\stackrel{a}{\rightarrow} \bullet$ that receives the atom a. If $\mathbb{A}$ containing $\mathsf{Lsn}[a]$ is realized, then a is available without protection in $\mathbb{A}$, i.e. a is **compromised**. We use listener strands to test whether atoms are safe secrets. Suppose a skeleton $\mathbb{A}'$ is the result of adding $\mathsf{Lsn}[a]$ to $\mathbb{A}$, and there is no homomorphism mapping $\mathbb{A}'$ to any realized $\mathbb{A}''$. Then A is safe in $\mathbb{A}$, as no execution described by $\mathbb{A}$ is compatible with a being compromised. Listener strands, lacking transmission nodes, need never precede anything else; we always let them be maximal in $\preceq_{\mathbb{A}}$.

If $\mathbb{A}, \mathbb{A}'$ are both realized, and differ only in which listener strands they contain, then we regard them as similar and write $\mathbb{A} \sim_{\mathsf{L}} \mathbb{A}'$. In this case, the skeleton $\mathbb{A}''$ that contains all listener strands from both $\mathbb{A}, \mathbb{A}'$ is also realized, and $\mathbb{A}'' \sim_{\mathsf{L}} \mathbb{A} \sim_{\mathsf{L}} \mathbb{A}'$. We will ignore differences between homomorphisms $H\colon \mathbb{A}_0 \mapsto \mathbb{A}$ and $H'\colon \mathbb{A}_0 \mapsto \mathbb{A}'$ that agree but have distinct, similar targets. Each may be extended by an embedding to yield the same homomorphism $H''\colon \mathbb{A}_0 \mapsto \mathbb{A}''$.

A homomorphism is a **contraction** if it identifies at least one pair of atoms and is surjective on nodes. A contraction replaces C with B in Fig. 1 to produce the Needham-Schroeder intended run.

Suppose that S is a set of encrypted messages and the atom $a \in \mathsf{unique}_{\mathbb{A}}$ originates at n_0. The pair of nodes n_0, n_1, where n_1 is a reception node, form an **outgoing test pair** for a and S iff all a's occurrences in $\mathsf{msg}(n_0)$ are within messages in set S, but a has at least one occurrence in n_1 outside the messages in S.[2] The second and fourth nodes on the responder strand, for instance, form an outgoing test pair for N_b and $S_0 = \{\{\!|A \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(B)}\}$, or for any $S_0' \supseteq S_0$. The set of keys used for outermost encryptions in any S is called $\mathsf{used}(S)$, i.e. $\mathsf{used}(S) = \{K : \exists t \,.\, \{\!|t|\!\}_K \in S\}$. So $\mathsf{used}(S_0) = \{\mathsf{ltk}(B)\}$.

The nodes m_0, m_1, with m_1 a transmission node, are an **outgoing transforming edge** for a, S if (1) they lie on the same regular strand $\ldots \Rightarrow^* m_0 \Rightarrow^+ m_1 \Rightarrow^* \ldots$; (2) a occurs in $\mathsf{msg}(m_0)$ but no earlier node; (3) a occurs outside S in $\mathsf{msg}(m_1)$ but not in any earlier node. In the Yahalom protocol, the second and third nodes of the server role are an outgoing transforming edge for N_b, S_0, although not for the larger set $S_0' = \{\{\!|A \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(B)}, \{\!|B \hat{} K \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}\}$. However, the second and third nodes of the initiator role are an outgoing transforming edge for N_b, S_0'.

Types of Search Step. There are two types of search steps, *outgoing steps* and *incoming steps*. The *outgoing* step states that each outgoing test pair n_0, n_1 must be solved, either by contracting atoms, or else by adding an outgoing transforming edge or a listener strand.

Outgoing test principle. *Let $H : \mathbb{A}_0 \mapsto \mathbb{A}_1$ 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 no outgoing transforming edge for a, S that precedes n_1, then, for some H'', $H = H'' \circ H'$ where either:*

1. *H' is a contraction; or*
2. *$H' : \mathbb{A}_0 \mapsto \mathbb{A}'$ 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 for $H(S)$. 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. Clause 1 is needed *only* in these two cases. Clause 3 uses the inverse K^{-1} because in public-key (asymmetric) algorithms, the adversary would use the inverse key K^{-1} to extract a from an occurrence within a message $\{\!|t|\!\}_K \in S$. We regard $\mathsf{pubk}(A), \mathsf{privk}(A)$ as inverses; symmetric keys are self-inverse.

[2] A message t_0 *occurs only within S in t_1* if, in the abstract syntax tree for t_1, every path to an occurrence of t_0 as a subterm traverses some member of S. A message t_0 *occurs outside S in t_1* if $t_0 \sqsubseteq t_1$ and t_0 does not occur only within S in t_1 [6]. In our terminology (Section 2), the K in $\{\!|t|\!\}_K$ is not an occurrence as a subterm.

The older outgoing test [10] lacked the set parameter S and applied (in effect) only to singleton S. The Yahalom analysis requires a non-singleton S. Only finitely many homomorphisms (to within isomorphism) can satisfy an instance of Clauses 1–3, because only finitely many atoms are mentioned in $\mathbb{A}_0$ and only finitely many transforming edges exist in one protocol. In particular, there is a finite set of most general homomorphisms. A set of homomorphisms $\{H_k\}_{k\leq j}$ is an **outgoing cohort** if, for some instance of Clauses 1–3, each H_k satisfies a clause, and for every H' satisfying one of those clauses, there is some $k \leq j$ and some H'' such that $H' = H'' \circ H_k$.

In a simple though not quite complete version, the *incoming* step states that if $\{|t|\}_K$ is received, either $K$is compromised or a regular strand transmitted it.

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 $\mathsf{Lsn}[K]$.*

We use Clause 1 only to prepare for an application of Clause 2, when n_1 is more general than a node in a role of the protocol. Again, there are finite sets $\{H_k\}_{k\leq j}$ that satisfy Clauses 1–3 in a most general way; we call them **incoming cohorts**. We call the skeletons $\{\mathbb{A}_k\}_{k\leq j}$ a cohort if each $H_k\colon \mathbb{A} \mapsto \mathbb{A}_k$ for some outgoing or incoming cohort $\{H_k\}_{k\leq j}$. In practice, for protocols that occur naturally, the size of the cohorts is very small, no more than four in the Yahalom protocol.

5 Yahalom: Shapes for the Responder

Suppose an execution contains a local run s_r of the responder's role as in the upper right column of Fig. 2. We assume the long term keys $\mathsf{ltk}(A), \mathsf{ltk}(B)$ are uncompromised, as no authentication can be achieved otherwise. Similarly, we assume the responder's nonce N_b to be fresh and unguessable.

So let the initial skeleton $\mathbb{A}_0$ consist of s_r, with $\mathsf{non}_{\mathbb{A}_0} = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$ and $\mathsf{unique}_{\mathbb{A}_0} = \{N_b\}$. What skeletons are shapes for $\mathbb{A}_0$? Or more precisely, for what realized skeletons $\mathbb{A}$ is there a shape $H\colon \mathbb{A}_0 \mapsto \mathbb{A}$?

We will find only one possibility, the skeleton $\mathbb{A}_4$ (Fig. 5). Any realized $\mathbb{A}$ containing any responder strand s'_r—with uncompromised long-term keys and a fresh nonce—has a subskeleton $\mathbb{A}'$ containing s'_r, with $J\colon \mathbb{A}_4 \mapsto \mathbb{A}'$. J is both nodewise injective and surjective, i.e. an isomorphism on nodes, although it may identify atoms. The portion of $\mathbb{A}$ containing s'_r resembles $\mathbb{A}_4$.

Transforming the Nonce. B chooses a fresh nonce N_b in node n_0 (see Fig. 3), and transmits it within the encrypted unit $\{|A \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(B)}$. In B's node n_2, it is received outside that unit, in the form $\{|N_b|\}_K$. So n_0, n_2 is an outgoing test pair for N_b, S_1 where$S_1 =$

$$\{\{|A \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(B)}\} \cup \{\{|B \hat{} K' \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(A)}\colon K' \text{ a key}\}.$$

1. The only outgoing transforming edges for N_b, S_1 lie on initiator strands. Unifying node 2 of the role with messages in S_1 shows that the parameters must be A, B, N_a, N_b, and some K'. We ask later whether $K' = K$.
2. Alternatively some decryption key may be compromised. Since $\mathsf{used}(S_1) = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$ contains symmetric (self-inverse) keys, this means we consider adding $\mathsf{Lsn}[\mathsf{ltk}(A)]$ or $\mathsf{Lsn}[\mathsf{ltk}(B)]$.

No contraction is relevant. Thus, these three embeddings—adding to $\mathbb{A}_0$ an initiator strand s_i, a listener strand $\mathsf{Lsn}[\mathsf{ltk}(A)]$, or one of the form $\mathsf{Lsn}[\mathsf{ltk}(B)]$—form an outgoing cohort. When adding s_i, we know that $n_0 \prec (s_i \downarrow 2) \Rightarrow (s_i \downarrow 3) \prec n_2$.

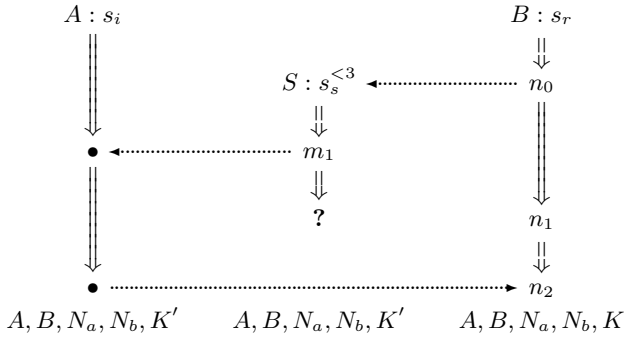

Fig. 3. Skeleton $\mathbb{A}_1$, with $\mathsf{non}_{\mathbb{A}_1} = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$ and $\mathsf{unique}_{\mathbb{A}_1} = \{N_b, K'\}$.

Since $\mathsf{non}_{\mathbb{A}_0} = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$, we also know that $\mathbb{A}_0 \cup \mathsf{Lsn}[\mathsf{ltk}(A)]$ and $\mathbb{A}_0 \cup \mathsf{Lsn}[\mathsf{ltk}(B)]$ are unrealizable. No bundle $\mathcal{B}$ can ever contain a listener strand for a value that originates nowhere. Thus, the embeddings of Case 2 are *dead* in the sense that no homomorphism from $\mathbb{A}_0$ to a realized skeleton can begin this way. Thus, every homomorphism from $\mathbb{A}_0$ to a realized skeleton factors through the embedding $\mathbb{A}_0 \mapsto \mathbb{A}_0 \cup \{s_i\}$.

We again have an outgoing test edge between n_0 and $s_i \downarrow 2$, for N_b, S_2 where

$$S_2 = \{\{\!|A \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(B)}\} \cup \{\{\!|B \hat{} K'' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)} \colon K'' \neq K'\}.$$

N_b originates only at n_0, where it occurs only within S_2; however, in $\mathsf{msg}(s_i \downarrow 2)$, N_b occurs outside S_2 in the form $\{\!|B \hat{} K' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}$.

3. The only outgoing transforming edges for N_b, S_2 lie on server strands s_s. Unifying node 1 of the role with messages in S_2 shows that the parameters must be A, B, N_a, N_b, and some K''. Since N_b must occur outside S_2 in $s_s \downarrow 2$, we have $K'' = K'$; so that the last parameter is K'. The last node $s_s \downarrow 3$ may not be included; we will write $s_s^{<3}$ for the initial segment of s_s.

4. Alternatively a decryption key in $\mathsf{used}(S_1) = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$ may be compromised. However, neither listener strand produces a live skeleton.

Thus, any homomorphism from $\mathbb{A}_0$ to a realized skeleton must factor through the embedding $\mathbb{A}_0 \mapsto \mathbb{A}_0 \cup s_r \cup s_s^{<3}$. We call this skeleton $\mathbb{A}_1$, shown in Fig 3, which also shows how the ordering relation extends. Since the server always provides a fresh session key, we also have $K' \in \mathsf{unique}_{\mathbb{A}_1}$.

Does $K' = K$? The server generated K' on strand s_s and delivers it to A on $s_i \downarrow 2$. B receives K on n_1, and on n_2 finds K also used to encrypt the nonce N_b. Must the keys K', K be the same, or could they be distinct?

Nodes n_0, n_2 form an outgoing test pair for N_b and the set

$$S_3 = \{\, \{|A \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(B)}, \{|B \hat{} K' \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(A)}, \{|N_b|\}_{K'} \,\}.$$

The resulting outgoing cohort consists of Cases 5–7:

5. Another server strand s'_s could receive N_b in its original form and transmit N_b and a new session key K'' as $\{|B \hat{} K'' \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(A)}$.
6. Under the contraction β that maps $K' \mapsto K$ and is elsewhere the identity, no new edge is needed, as $\{|N_b|\}_{K'} \cdot \beta = \{|N_b|\}_K \cdot \beta$.
7. $\mathsf{used}(S_3) = \{\mathsf{ltk}(A), \mathsf{ltk}(B), K'\}$. Although adding $\mathsf{Lsn}[\mathsf{ltk}(A)]$ and $\mathsf{Lsn}[\mathsf{ltk}(B)]$ lead to dead skeletons, perhaps adding $\mathsf{Lsn}[K']$ does not, i.e. K' may become compromised.

However, we can prune Case 5, because K'' is not usefully different from K'. The adversary cannot use messages transmitted by s'_s differently from the messages transmitted by the existing s_s. Discarding Case 5, there are two live possibilities: either $K' = K$ or else K' becomes compromised. We consider Case 7 next.

Case 7: K' becomes compromised. Consider the skeleton $\mathbb{A}_1 \cup \mathsf{Lsn}[K']$. K' originates uniquely at m_1, so $m_1, (\mathsf{Lsn}[K'] \downarrow 1)$ is an outgoing transformed pair for K' and $S_4 = \{\, \{|B \hat{} K' \hat{} N_a \hat{} N_b|\}_{\mathsf{ltk}(A)}, \{|A \hat{} K'|\}_{\mathsf{ltk}(B)} \,\}$. Thus, some case in the cohort 8–9 must hold:

8. Some role $\mathsf{Init}, \mathsf{Resp}, \mathsf{Serv}$ provides a transforming edge for K', S_4. However, no Yahalom role retransmits it as a subterm of any new message. The initiator uses K' to encrypt a message, but in our model, this discloses nothing. For finer models, see e.g. [3, 5].
9. One of the keys that protects K' in S_4, i.e. a key $K_0 \in \mathsf{used}(S_4)$, becomes compromised; but $\mathsf{used}(S_4) = \{\, \mathsf{ltk}(A), \mathsf{ltk}(B) \,\}$.

So neither Case 8 nor Case 9 is possible. We discard Case 7, as the whole cohort 8–9 is unrealizable or "dead."

Hence, all homomorphisms to realized skeletons must factor through Case 6. Let $\mathbb{A}_2 = \mathbb{A}_1 \cdot \beta$ be the result of replacing K' by K wherever mentioned in $\mathbb{A}_1$. If any homomorphism $H \colon \mathbb{A}_0 \mapsto \mathbb{A}'$ has $\mathbb{A}'$ realized, then H factors through the embedding $\mathbb{A}_0 \mapsto \mathbb{A}_2$.

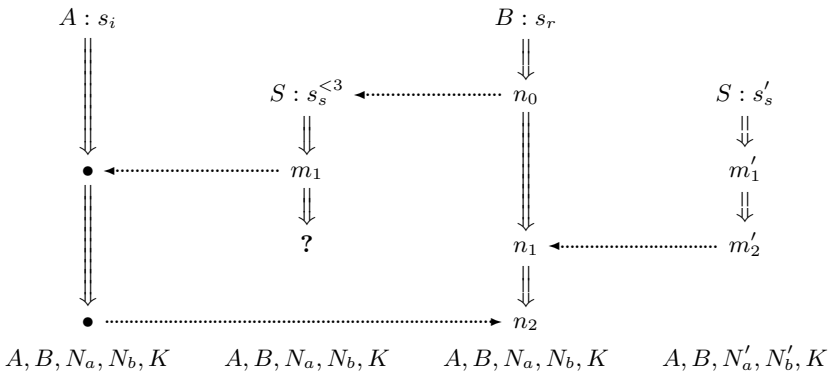

Fig. 4. $\mathbb{A}_3$, with $\mathsf{non}_{\mathbb{A}_3} = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$ and $\mathsf{unique}_{\mathbb{A}_3} = \{N_b, K\}$.

***B*'s Source for *K*.** The responder B receives $\{\!|A \hat{} K|\!\}_{\mathsf{ltk}(B)}$ on node n_1. We apply the *Incoming Test Principle*, with cohort:

10. A server strand s'_s, with parameters A, B, K, transmits $\{\!|A \hat{} K|\!\}_{\mathsf{ltk}(B)}$; possibly different nonces appear in s'_s. The embedding yields $\mathbb{A}_3$ in Fig. 4.
11. Alternatively, $\mathsf{ltk}(B)$ has been compromised and $\{\!|A \hat{} K|\!\}_{\mathsf{ltk}(B)}$ is generated by the adversary. However, $\mathsf{ltk}(B) \in \mathsf{non}_{\mathbb{A}_2}$, excluding this case.

$\mathbb{A}_3$ is not a skeleton because of an anomaly, however. $K \in \mathsf{unique}_{\mathbb{A}_3}$ is intended to originate at just one node, but in fact originates at both m_1 and m'_1. Therefore,

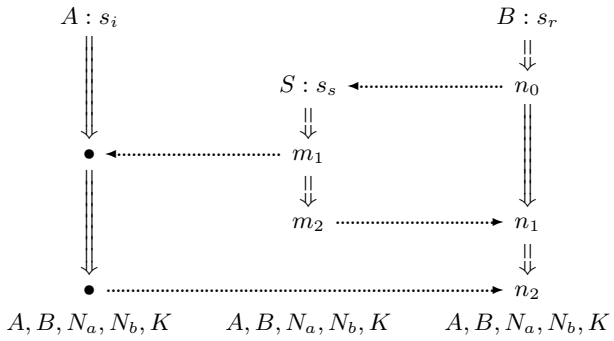

Fig. 5. Skeleton $\mathbb{A}_4$, with $\mathsf{non}_{\mathbb{A}_4} = \{\mathsf{ltk}(A), \mathsf{ltk}(B)\}$ and $\mathsf{unique}_{\mathbb{A}_4} = \{N_b, K\}$.

in any skeleton obtained by a homomorphism $H = [\phi, \alpha]$ jointly from the union $\mathbb{A}_2 \cup \{s'_s\} = \mathbb{A}_3$, necessarily $\phi(m_1) = \phi(m'_1)$, equating the strands s_s and s'_s. H

must then factor through skeleton $\mathbb{A}_4$ (Fig. 5), where consequently $N_a \cdot \alpha = N'_a \cdot \alpha$ and $N_b \cdot \alpha = N'_b \cdot \alpha$, and the height of $\phi(s_s)$ is 3.

Skeleton $\mathbb{A}_4$ is *realized*: every message received is sent, even without adversary activity. Moreover, $\mathbb{A}_4$ is a *shape*. First, if we leave out any nodes, then either B's original 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 be realized if A's parameter N_b were altered to some N'_b. Since all homomorphisms from $\mathbb{A}_0$ to realized skeletons factor through $\mathbb{A}_4$, it is the only shape for $\mathbb{A}_0$.

6 Search Strategy

The goal of CPSA is defined using the following terms:

$\mathsf{step}(\mathbb{A}, C)$, which holds if the finite set C of skeletons is 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 4 imply that the tests and their cohorts may be used in any order, while still finding all shapes.

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

$\mathsf{min\_real}_{\mathbb{A}_0}(\mathbb{A}')$, which is defined if $\mathbb{A}'$ is realized. Its value is the finite, non-empty set of shapes $\mathbb{A}$ 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).

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:

$\mathsf{dead}(\mathbb{A})$ means $\mathbb{A}$ cannot be realized, i.e. there is no realized $\mathbb{A}'$ with $H \colon \mathbb{A} \mapsto \mathbb{A}'$. $\mathsf{Dead}(\mathbb{A})$ 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. Condition (1) was used repeatedly and condition (3) was used to discard Case 7, as the cohort 8–9 led only to dead skeletons.

$\mathsf{redundant\_strand}(\mathbb{A})$ 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. We discarded a redundant strand in 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 a cohort, updates the step relation, and then returns the cohort (assuming $\mathsf{step\_applies}(\mathbb{A})$ is true).

$\mathsf{targets}(\boldsymbol{H}) = \{\mathbb{A}_k \colon k \leq j\}$, if $\boldsymbol{H}$ 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 reached, because completeness [6] ensures that when $\mathbb{A}$ is not realized, then some authentication test step applies.

```
F := {A_0};   shapes_found := ∅;   seen := F;
while F ≠ ∅ begin
      A := select(F);   F := F \ {A};
      if realized(A)
        then shapes_found := shapes_found ∪ min_real_{A_0}(A)
      else if redundant_strand(A) then skip
      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_found
```

Fig. 6. CPSA Search Algorithm

7 Implementing CPSA

We discuss here three aspects of the CPSA implementation. They are: finding candidate transforming edges 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 transforming edges. When CPSA reads a protocol description in its input format, 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 edge $m_0 \Rightarrow^+ m_1$. 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 $m_0 \Rightarrow^+ m_1$ is a successful candidate. If α contracts atoms, then we apply the Outgoing Test Principle twice, once to apply this contraction, and once to add the instance of $m_0 \Rightarrow^+ m_1$.[3] We also check whether a contraction eliminates the outgoing test edge entirely, as in Case 6.

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 use a trick we call the "forwards-then-backwards" technique. CPSA plans a sequence of applications of the outgoing test until no further transforming edge is found, as in Yahalom cases 3 and 1. It follows the transmission of the

[3] This is the only aspect of the authentication test search that does not occur in the Yahalom analysis.

uniquely originating value—N_b in that case—forwards. Newly introduced atoms like K' are implicitly universal. Originally, N_b occurs only in $\{\!|A \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(B)}$; after a server strand it also occurs in $\{\!|B \hat{} K' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)}$. After an initiator strand, no other transforming edges can succeed.

CPSA uses the sets in the opposite order. The set $S_1 = \{\{\!|A \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(B)}\} \cup \{\{\!|B \hat{} K' \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(A)} : K' \text{ a key}\}$ is used first to introduce the initiator transforming edge. Then the smaller set $\{\{\!|A \hat{} N_a \hat{} N_b|\!\}_{\mathsf{ltk}(B)}\}$ is used to introduce the (earlier) server transforming edge.[4]

The forwards-then-backwards technique suggested CPSA's representation for the sets S. These sets are not necessarily finite; S_1 e.g. is not. The family is closed under union and set difference. The primitive members are singletons $\{t_0\}$ and sets that represent all the instances of a term t_1 as some of t_1's parameters vary. Thus, we can represent all candidate sets are as finite unions and differences of values of the form $\lambda \boldsymbol{v} \,.\, t$, where the vector $\boldsymbol{v}$ binds 0 or more atoms in t. Completeness requires only sets S representable in this form.

This representation fits also nicely with our use of unification to provide an extremely focused search, leading to good runtimes on a variety of protocols. Samples run on a Thinkpad X31, with a 1.4 GHz Pentium M processor and 1 GB store, under Linux, are shown in Fig. 7. CPSA is implemented in OCaml.

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

Fig. 7. Protocols with CPSA runtimes

Future work. The soundness of the search algorithm does not require the bare-bones Dolev-Yao model used here. One can augment CPSA with Diffie-Hellman operations, as studied in [11]. One can 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 terms in general, at the cost of using more sophisticated methods to check whether skeletons are realized (e.g. [15]). The skeletons-and-homomorphisms approach may remain useful in a cryptographic, asymptotic probabilistic context.

Acknowledgments. We thank Lenore Zuck and John D. Ramsdell for their comments. Larry Paulson suggested the Yahalom protocol as a challenge.

[4] The cleverer set S_2 we used in Case 3 is an optimization. To ensure that the server and initiator agree on the session key, CPSA uses instead a cohort similar to Cases 5–7.

References

1. Martín Abadi and Bruno Blanchet. Analyzing security protocols with secrecy types and logic programs. *Journal of the ACM*, 52(1):102–146, January 2005.
2. Roberto M. Amadio and Denis Lugiez. On the reachability problem in cryptographic protocols. In *Concur*, number 1877 in LNCS, pages 380–394, 2000.
3. Michael Backes and Birgit Pfitzmann. Relating cryptographic and symbolic key secrecy. In *Proceedings, 26th IEEE Symposium on Security and Privacy*, May 2005. Extended version, `http://eprint.iacr.org/2004/300`.
4. Bruno Blanchet and Andreas Podelski. Verification of cryptographic protocols: Tagging enforces termination. In *Foundations of Software Science and Computation Structures*, LNCS, pages 136–152, April 2003.
5. Ran Canetti and Jonathan Herzog. Universally composable symbolic analysis of mutual authentication and key exchange protocols. In *Proceedings, Theory of Cryptography Conference (TCC)*, March 2006.
6. Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Searching for shapes in cryptographic protocols (extended version). `URL:http://eprint.iacr.org/2006/435`, November 2006.
7. Daniel Dolev and Andrew Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.
8. 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.
9. Andrew D. Gordon and Alan Jeffrey. Types and effects for asymmetric cryptographic protocols. *Journal of Computer Security*, 12(3/4):435–484, 2003.
10. Joshua D. Guttman and F. Javier Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 283(2):333–380, June 2002. Conference version appeared in *IEEE Symposium on Security and Privacy*, May 2000.
11. Jonathan C. Herzog. The Diffie-Hellman key-agreement scheme in the strand-space model. In *16th Computer Security Foundations Workshop*, pages 234–247, Asilomar, CA, June 2003. IEEE CS Press.
12. ITU. Message sequence chart (MSC). Recommendation Z.120, 1999.
13. Leslie Lamport. Time, clocks and the ordering of events in a distributed system. *CACM*, 21(7):558–565, 1978.
14. 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.
15. 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.
16. Roger Needham and Michael Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), 1978.
17. Lawrence C. Paulson. Relations between secrets: Two formal analyses of the Yahalom protocol. *Journal of Computer Security*, 2001.
18. 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.
19. 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.