Shapes: Surveying Crypto Protocol Runs*

Joshua D. Guttman

January 12, 2010

Abstract

Given a cryptographic protocol, and some assumptions, can we present *everything that can happen*, subject to these assumptions? The assumptions may include: (i) some behavior assumed to have occurred, (ii) some keys assumed to be uncompromised, and (iii) some values assumed to have been freshly chosen. An object representing these types of information is called a *skeleton*.

The *shapes* for a skeleton $\mathbb{A}$ are the minimal, essentially different executions that are compatible with the assumptions in $\mathbb{A}$. The set of shapes for an $\mathbb{A}$ is frequently but not always finite. Given a finite set of shapes for $\mathbb{A}$, it is evident whether a security goal such as authentication or confidentiality holds for $\mathbb{A}$.

In this paper, we describe a *search* that finds the shapes, starting from a protocol and a skeleton $\mathbb{A}$. The search is driven by the challenge-response patterns formalized in the strand space *authentication tests*.

We define the framework of skeletons and homomorphisms within which the search proceeds. We justify the search steps by showing: If a shape is compatible with the skeleton from which a step starts, then it is still compatible with at least one skeleton resulting from the step. Thus, if the search terminates, it will report every shape. We also show that the shapes are accessible via these search steps: Every shape can be reached in finitely many search steps.

1 Initial Examples

In this paper, we will develop a search technique for finding out all the minimal, essentially different executions that are possible in a protocol, starting from some initial assumptions about behavior. We use this method to determine counterexamples to authentication and confidentiality properties, when they are not true. Alternatively, we can establish these properties, when they hold, in the (common but not universal) case in which the search terminates.

*This work partly supported by the National Science Foundation, grant number CNS-0952287. Preliminary versions of some of this material appeared in [6, 5]. That underlying work was joint with Shaddin Doghmi and Javier Thayer. Parts of that work was funded by MITRE-Sponsored Research, and parts were funded by the National Security Agency. Address: guttman@{wpi.edu,mitre.org}.

We will start by carrying out several intuitive analyses, using Blanchet's simple example protocol [2]. The remainder of the chapter will formalize these intuitive analyses and justify the method. Blanchet's protocol, which we will call SEP, requires an initiator A to generate a fresh symmetric key k, sign and encrypt it for a chosen responder B, and await reception of a message $\{|s|\}_k$.[1] On the other side, any responder B will await a message containing a signed and encrypted k, at which point it will select a secret s to transmit encrypted with k. We represent these two kinds of behavior schematically as shown in Fig. 1. We call a finite sequence of sends and receives a *strand*, so the behavior

Figure 1: SEP: Blanchet's Simple Example Protocol

of the initiator or responder in a single local session is a strand. Strands are written either vertically or horizontally as sequences of nodes connected by double arrows $\bullet \Rightarrow \bullet$ etc. We also write $n_0 \Rightarrow^+ n_1$ when n_1 follows n_0 on the same strand, although possibly not immediately.

1.1 A's Point of View

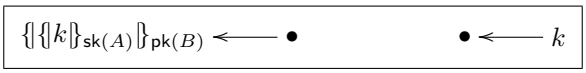
We start by exploring A's point of view. We consider the assumption that A has engaged in one or more steps of a local session of SEP, and we consider what other behavior must occur in any possible execution containing this behavior.

This is the *point-of-view principle*: A's point of view is that he *knows* that he engaged in these steps of his local session. He would like to *infer* as much as possible about what other behavior must have occurred, or could not have occurred.

The point of view principle is central to protocol analysis. It is largely the activity of exploring what behaviors are possible, given some initially assumed behavior. This initial assumed behavior is usually a run of one principal. In that case, we regard the analysis as telling us what behavior must have occurred in the distributed system, from the *point of view* of that principal.

Secrecy of the session key k. Suppose that an initiator A has executed at least the first node of a session, transmitting a session key k within a message $\{|\{|k|\}_{\mathsf{sk}(A)}|\}_{\mathsf{pk}(B)}$. Is A guaranteed that an adversary can never obtain the value k in a form protected by no encryption? The answer is *no*, in at least two cases.

[1] Since we frequently refer to sets, we reserve $\{vs\}$ for set formation. We will write any encryption in the form $\{|p|\}_K$ where p is the plaintext and K is the key used to prepare it. We will not distinguish symmetric and asymmetric encryption notationally, but let the key determine which applies. Thus, when an encryption key and its inverse decryption key are equal, i.e. $K = K^{-1}$, then $\{|p|\}_K$ is a symmetric encryption.



$\mathsf{non} = \{\mathsf{pk}(B)^{-1}\}$ $\quad$ $\mathsf{unique} = \{k\}$

Figure 2: Skeleton $\mathbb{A}_0$: Disclosure of k?

1. When the key generator chosing k lacks randomness: An adversary may then generate the candidate keys and (possibly) test which was sent.

   Alternatively, the way k was chosen may ensure that it is fresh and unguessable; we use the term *uniquely originating* for such a k.

   A originates k for this transmission, and any other place it is sent or received must then derive in an understandable way from this transmission or its later transformed forms. An adversary's generate-and-test would be a separate point of origination for the same value. Likewise, if a protocol participant were, by bad luck, to generate the same k for another run, that would be an additional point of origination for k. A reasonable cryptographic implementation of SEP should ensure that these events are of negligible likelihood in some suitable sense.

2. When B's private decryption key $\mathsf{pk}(B)^{-1}$ is compromised: An adversary can then extract the signed unit from $\{|\{|k|\}_{\mathsf{sk}(A)}|\}_{\mathsf{pk}(B)}$, check the signature, and extract k.

   It is irrelevant whether B does this ("B is dishonest") or whether B's secret $\mathsf{pk}(B)^{-1}$ has fallen into the hands of some malicious party. In either case, B's private decryption key has been used in a way that is not stipulated in the protocol definition. Thus, we say that a key is *uncompromised* if it is used only in accordance with the protocol under analysis.

   In our formalism, a key used contrary to the stipulations of the protocol must always originate. Thus, we call an uncompromised key *non-originating*.

A strand of the protocol is called a *regular* strand. Thus, all local behaviors divide into regular strands and adversary behaviors. We sometimes say that a principal A is *regular* if its private keys are used only in regular strands.

Is there any third way that an adversary could obtain the key k?

To answer this question, we will carry out an experiment. We will start with a diagram (Fig. 2) that represents a transmission of $\{|\{|k|\}_{\mathsf{sk}(A)}|\}_{\mathsf{pk}(B)}$ and the reception of k, somehow shorn of all cryptographic protection. We call a node like the right hand node of Fig. 2 a *listener* node, since it listens, and hears the value k, thereby witnessing that k has been disclosed. The diagram also incorporates the assumption that neither case 1 nor case 2 above applies, i.e. k is uniquely originating and $\mathsf{pk}(B)^{-1}$ is non-originating. We would like to

discover whether we can embed this diagram into a richer or more informative diagram that represents a possible execution.

If any enriched version of $\mathbb{A}_0$ can occur, then there must be some earliest point at which k is transmitted outside of the cryptographic protection of $\{|\{|k|\}_{\mathsf{sk}(A)}|\}_{\mathsf{pk}(B)}$. If the adversary could use $\mathsf{pk}(B)^{-1}$, this could occur via an adversary decryption, but the assumption $\mathsf{pk}(B)^{-1} \in \mathsf{non}$ excludes this. If the adversary could be lucky enough to re-originate the same k, then this re-origination would be an earliest transmission unprotected by $\{|\{|k|\}_{\mathsf{sk}(A)}|\}_{\mathsf{pk}(B)}$. The assumption $\mathsf{unique} = \{k\}$ excludes this. Thus, any earliest transmission of k outside the form $\{|\{|k|\}_{\mathsf{sk}(A)}|\}_{\mathsf{pk}(B)}$ must be caused by some participant executing a strand of the protocol.

However, when we examine Fig. 1, we see that a symmetric key is received by a participant only on the first node of a responder strand. This key however is not retransmitted; instead, k is used to encrypt the payload s, and the ciphertext $\{|s|\}_k$ can never lead to the disclosure of k. A principal that already knows k can use it to obtain s, but a principal that does not yet have information about k cannot obtain k from $\{|s|\}_k$. If an adversary can recognize s and has a hypothesis about the value of k, then it can use the message $\{|s|\}_k$ to *check* the hypothesis. However, we will be concerned only with full disclosure, not with a subtler notion of secrecy that resists hypothesis checking.

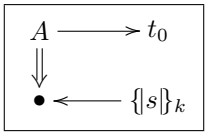
We have now exhausted all the possibilities. $\mathbb{A}_0$ is a dead end. No enrichment of $\mathbb{A}_0$ can be an execution that can possibly occur. We call it a *dead skeleton.*

In reaching this conclusion, we have used a principle that will be central to the search for shapes:

Principle 1.1 (The Nonce Test) *Suppose that $c \in \mathsf{unique}$, and c is found in some message received in a skeleton $\mathbb{A}$ at a node n_1. Moreover, suppose that, in the message of n_1, c is found outside all of a number of encrypted forms $\{|t_1|\}_{K_1}, \ldots, \{|t_j|\}_{K_j}$. Then in any enrichment $\mathbb{B}$ of $\mathbb{A}$ such that $\mathbb{B}$ is a possible execution, either:*

1. *One of the matching decryption keys ${K_i}^{-1}$ is disclosed before n_1 occurs, so that c could be extracted by the adversary; or else*
2. *Some regular strand contains a node m_1 in which c is transmitted outside the forms $\{|t_1|\}_{K_1}, \ldots, \{|t_j|\}_{K_j}$, but in all previous nodes $m_0 \Rightarrow^+ m_1$, c was found (if at all) only within the plaintexts $t_1 \ldots t_j$. Moreover, m_1 occurs before n_1.*

This says that if c is extracted from the encrypted forms, then, in any possible execution, either the adversary can do so (Case 1), which we witness by adding a listener node for a decryption key ${K_i}^{-1}$; or else some regular strand has done so (Case 2). We have just applied Principle 1.1 in the case where $c = k$, $j = 1$, $K_1 = \mathsf{pk}(B)$, and $t_1 = \{|k|\}_{\mathsf{sk}(A)}$. In this application, Case 1 was excluded by the assumption $\mathsf{pk}(B)^{-1} \in \mathsf{non}$. The behavior described in Case 2 has no instance in common with any protocol behavior in Fig. 1. Hence the dead end.



$\mathsf{non} = \{\mathsf{pk}(B)^{-1}\}$ $\quad$ $\mathsf{unique} = \{k\}$

Figure 3: Skeleton $\mathbb{B}$; t_0 is $\{\!|\{\!|k|\!\}_{\mathsf{sk}(A)}|\!\}_{\mathsf{pk}(B)}$

We use the list $\{\!|t_1|\!\}_{K_1}, \ldots, \{\!|t_j|\!\}_{K_j}$ because a protocol may re-use a nonce several times. After a nonce has been transmitted inside the encryption $\{\!|t_1|\!\}_{K_1}$ and received back inside the encryption $\{\!|t_2|\!\}_{K_2}$, it may be retransmitted inside the encryption $\{\!|t_3|\!\}_{K_3}$. If it is ever received back in some new form $\{\!|t_4|\!\}_{K_4}$, then that transformation needs an explanation of one of the two forms mentioned in Principle 1.1. If it is ever received back with no further encryption, then it can no longer be reused in this way.

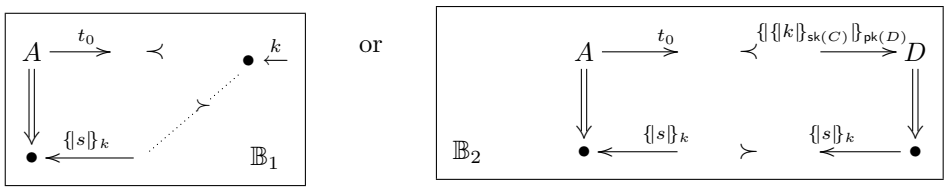
***A*'s Authentication Guarantee.** Suppose that an initiator has executed a local session of its role in SEP. What forms are possible for the execution as a whole global behavior? In exploring this question, we will make the same assumptions about non and unique. Thus, we represent this graphically in the form shown in Fig. 3, where for brevity we write $t_0 = \{\!|\{\!|k|\!\}_{\mathsf{sk}(A)}|\!\}_{\mathsf{pk}(B)}$. We again ask what explanations could exist for the various nodes, i.e. what enrichment could elaborate $\mathbb{B}$ into a skeleton that represents a possible execution. The first node requires no explanation, since A transmits $\{\!|\{\!|k|\!\}_{\mathsf{sk}(A)}|\!\}_{\mathsf{pk}(B)}$ just as the protocol indicates.

By contrast, the second node, A's reception of $\{\!|s|\!\}_k$, does require an explanation: Where did $\{\!|s|\!\}_k$ come from?

1. Possibly k is disclosed to the adversary, who then prepared the message $\{\!|s|\!\}_k$. We may test this candidate explanation by adding a listener node to witness disclosure of the encryption key k.

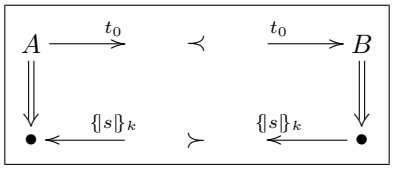
2. Alternatively, we may add a strand of the protocol, including a node that transmits $\{\!|s|\!\}_k$. As is evident from Fig. 1, this must be the second node of a responder strand. However, what values are possible for the other parameters of the strand, i.e. the names of the initiator and responder in this session? We will postpone the question by choosing new, unconstrained values C, D.

This leads us to the two descendants of $\mathbb{B}$, shown as $\mathbb{B}_1, \mathbb{B}_2$ in Fig. 4. We may now immediately exclude $\mathbb{B}_1$. It must be a dead end, because it is an enrichment of $\mathbb{A}_0$ in Fig. 2. If any enrichment of $\mathbb{B}_1$ were a possible execution, then it would be the enrichment of an enrichment of $\mathbb{A}_0$, and—since the composition of enrichments is an enrichment—some enrichment of $\mathbb{A}_0$ would be a possible execution.



$\mathsf{non} = \{\mathsf{pk}(B)^{-1}\}$ $\quad$ $\mathsf{unique} = \{k\}$

Figure 4: Analysis of $\mathbb{B}$, Step 1; t_0 is $\{|\{|k|\}_{\mathsf{sk}(A)}|\}_{\mathsf{pk}(B)}$



$\mathsf{non} = \{\mathsf{pk}(B)^{-1}\}$ $\quad$ $\mathsf{unique} = \{k\}$

Figure 5: Analysis of $\mathbb{B}$, Step 2: Its shape $\mathbb{B}_{21}$

Turning to $\mathbb{B}_2$, it has an unexplained node, the upper right node n_D receiving $\{|\{|k|\}_{\mathsf{sk}(C)}|\}_{\mathsf{pk}(D)}$. If it happens that $C = A$ and $D = B$, then nothing further need be done.

Otherwise, we may apply Principle 1.1. The value k, having been previously observed only in the form t_0, is now received on n_D in a different form, namely $\{|\{|k|\}_{\mathsf{sk}(C)}|\}_{\mathsf{pk}(D)}$. Since $\mathsf{pk}(B)^{-1} \in \mathsf{non}$, case 1 does not apply. We must thus have a regular strand that receives k only within the encrypted form t_0 and retransmits it outside of t_0. However, in analyzing $\mathbb{A}_0$, we have already seen that the protocol contains no such strand.

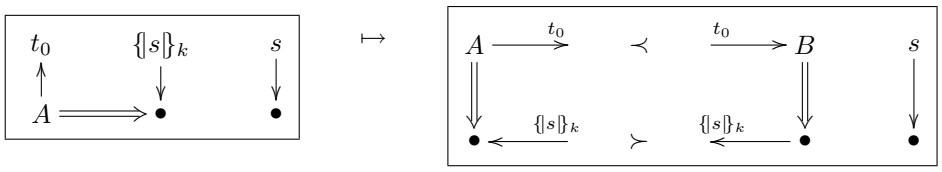
Thus, we are left with the single case of $\mathbb{B}_2$ in which $C = A$ and $D = B$, which is the desired execution $\mathbb{B}_{21}$ shown in Fig. 5. The index 21 is meant to indicate the path along which it was encountered, as the sole child of $\mathbb{B}_2$, which is itself the rightmost child of $\mathbb{B}$. $\mathbb{B}_{21}$ is the sole *shape* for $\mathbb{B}$: Any execution compatible with $\mathbb{B}$ must contain at least the behavior shown in $\mathbb{B}_{21}$.

We have made use of two additional principles in this analysis. One asserts that death persists; the other concerns the origin of encrypted messages.

Principle 1.2 *If a skeleton $\mathbb{A}$ is dead, then so is any enrichment $\mathbb{B}$ of $\mathbb{A}$.*

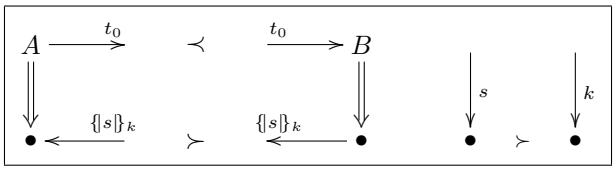
We applied Principle 1.2 to discard $\mathbb{B}_1$.

Principle 1.3 (The Encryption Test, 1) *Suppose that $\{|t|\}_K$ is found in some message received in a skeleton $\mathbb{A}$ at a node n_1. Then in any enrichment $\mathbb{B}$ of $\mathbb{A}$ such that $\mathbb{B}$ is a possible execution, either:*



$\mathsf{non} = \{\mathsf{pk}(B)^{-1}\}$ $\quad$ $\mathsf{unique} = \{k, s\}$

Figure 6: Skeletons $\mathbb{C}$ and $\mathbb{C}_{21}$



$\mathsf{non} = \{\mathsf{pk}(B)^{-1}\}$ $\quad$ $\mathsf{unique} = \{k, s\}$

Figure 7: Dead skeleton $\mathbb{C}_{211}$

1. *The encryption key K is disclosed before n_1 occurs, so that the adversary could construct $\{|t|\}_K$ from t; or else*

2. *Some regular strand contains a node m_1 in which $\{|t|\}_K$ is transmitted, but no previous node $m_0 \Rightarrow^+ m_1$ contains $\{|t|\}_K$. Moreover, m_1 occurs before n_1.*

We applied Principle 1.3 to construct skeletons $\mathbb{B}_1, \mathbb{B}_2$, using the instance $t = s$ and $K = k$. Case 1 furnished $\mathbb{B}_1$ and Case 2 yielded $\mathbb{B}_2$. The node n_1 is the later (reception) node of $\mathbb{B}$, and m_1 is the lower right transmission node in $\mathbb{B}_2$.

We will strengthen Principle 1.3 and combine it in a single form with Principle 1.1, resulting in the Authentication Test Principle, Theorem 5.5 of Section 5.

Secrecy of s. Can A be sure that the value s remains a secret between A and B? To test this, we start with an expansion of skeleton $\mathbb{B}$ in which there is also a listener node that observes s shorn of all cryptographic protection, as shown in the left portion of Fig. 6. The question is only relevant if s is assumed fresh and unguessable. $\mathbb{C}$ is an enrichment of $\mathbb{B}$. Every execution enriching $\mathbb{B}$ must contain at least the structure we found in $\mathbb{B}_{21}$, and it must also contain the listener node for s. Thus, it must be an enrichment of $\mathbb{C}_{21}$. Now, at this point we can apply Principle 1.1, instantiating $c = s$ and node n_1 being the listener at the lower right. The index $j = 1$, and the encrypted form containing s is $\{|s|\}_k$. Since k is a symmetric session key, $k^{-1} = k$. Since no regular strand of SEP receives a value encrypted by a symmetric key and retransmits that value in any other form, Case 2 of the principle is vacuous. Thus, we add a listener

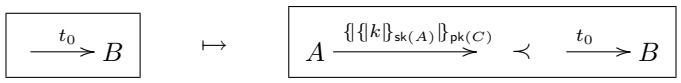


Figure 8: Skeleton $\mathbb{D}$: B's Point of View, and its shape $\mathbb{D}_1$

node for k, witnessing for its disclosure, obtaining $\mathbb{C}_{211}$ in Fig. 7. $\mathbb{C}_{211}$ is dead as a consequence of Principle 1.2, since $\mathbb{C}_{211}$ certainly enriches the dead skeleton $\mathbb{A}_0$ in Fig. 2.

Thus, SEP fulfills its goals, from the point of view of an initiator A.

In the step from $\mathbb{C}$ to $\mathbb{C}_{21}$, we used an additional principle:

Principle 1.4 *Suppose that $\mathbb{B}$ has the shapes $\mathbb{S}_1, \ldots, \mathbb{S}_i$. If $\mathbb{C}$ enriches $\mathbb{B}$, then every execution enriching $\mathbb{C}$ is an enrichment of some $\mathbb{S}_j$, where $1 \leq j \leq i$.*

Since $\mathbb{B}$ had the single shape $\mathbb{C}_{21}$, we applied Principle 1.4 with $i = 1$ and $\mathbb{S}_1 = \mathbb{B}_{21}$, allowing us to jump right from $\mathbb{C}$ to $\mathbb{C}_{21}$. We could also have reconstructed its contents using several applications of the other principles.

1.2 B's Point of View

The story is quite different when we turn to the point of view of a responder B.

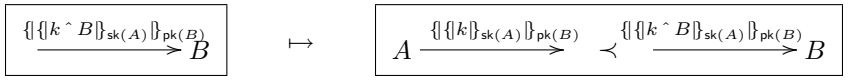
B's Authentication Guarantee. Suppose that a responder B has received a message of the form $t_0 = \{\!|\{\!|k|\!\}_{\mathsf{sk}(A)}|\!\}_{\mathsf{pk}(B)}$. Assuming now, in skeleton $\mathbb{D}$ of Fig. 8, that both A's private signature key $\mathsf{sk}(A)$ and B's private decryption key $\mathsf{pk}(B)^{-1}$ are non-originating, what else must have happened in any enrichment of $\mathbb{D}$ that is a possible execution? We may try to apply Principle 1.3 again, where the encrypted unit $\{\!|t|\!\}_K$ is $t_0 = \{\!|\{\!|k|\!\}_{\mathsf{sk}(A)}|\!\}_{\mathsf{pk}(B)}$. However, Case 1 then requires only that the public encryption key $\mathsf{pk}(B)$ of B is available to the adversary, from which we learn nothing.

We may more profitably apply Principle 1.3 by taking the encrypted unit $\{\!|t|\!\}_K$ to be $\{\!|k|\!\}_{\mathsf{sk}(A)}$. Since the key $\mathsf{sk}(A)$ is non-originating, Case 1 is vacuous. Thus, every possible execution must include an enrichment with a regular node producing $\{\!|k|\!\}_{\mathsf{sk}(A)}$. By Fig. 1, this must be the first node of an initiator strand. We know that the parameter representing the initiator's name is A, and the parameter representing the session key has the value k. However, we know nothing about the remaining parameter appearing in an initiator's first node, i.e. the name of the intended responder. Since this value is unconstrained, we fill it in with some new C, thus obtaining the skeleton $\mathbb{D}_1$.

Unfortunately, we cannot collect any more information about the parameter C, unlike our situation in skeleton $\mathbb{B}_2$ (Fig. 4). $\mathbb{D}_1$ contains all of the regular behavior needed for an execution. It is the sole shape for $\mathbb{D}$.

Nothing says that C's decryption key is uncompromised, so the adversary can decrypt the outer layer, using the public key $\mathsf{pk}(B)$ to re-encrypt $\{\!|k|\!\}_{\mathsf{sk}(A)}$

Figure 9: SEPC: the Simple Example Protocol Corrected



$\mathsf{sk}(A), \mathsf{pk}(B)^{-1} \in \mathsf{non}$

Figure 10: Skeleton $\mathbb{E}$: B's Point of View, and its shape $\mathbb{E}_1$

in the desired form. Evidently, the session key k may also be disclosed in this process. Thus, in SEP, a responder B does get a guarantee that A initiated a session with key k. However, since A may have chosen a compromised party C as partner for that conversation, B cannot count on much, certainly not that k, or any s encrypted with k, will remain confidential.

1.3 Correcting SEP

Principles 1.1 and 1.3 are central to protocol design [8] as well as to protocol analysis [6]. In SEP, our analysis of B's guarantee applied Principle 1.3 to the critical term $\{|k|\}_{\mathsf{sk}(A)}$. Since this term involves only the two parameters k, A, evidently this cannot force agreement on a particular responder. However, to force A to agree with B on the responder, it suffices to add B's name to this critical term. The resulting protocol SEPC takes the form given in Fig. 9. B's authentication result is shown in Fig. 10.

If we now add to $\mathbb{E}_1$ a listener node for k, and assume k uniquely originating, the resulting skeleton is an enrichment of $\mathbb{A}_0$. Principle 1.2 thus entails that k cannot be disclosed. When we extend B's strand to include its second node, transmitting $\{|s|\}_k$, it will also lead to the conclusion that s is undisclosed.

Our correction of SEP is tighter or more minimal than Blanchet's [2], where the signed unit $\{|k \hat{} A \hat{} B|\}_{\mathsf{sk}(A)}$ is used. The occurrence of A is unnecessary here. Principle 1.3 helped identify exactly the parameters that are needed in a protocol refinement.

1.4 Goals of this Chapter

So far, we have used Blanchet's Simple Example Protocol as an example to illustrate the idea that, from a particular starting point, one can find all of the minimal, essentially different things that can happen, compatible with that

starting point. We call the minimal, essentially different executions compatible with a starting point $\mathbb{A}$ its *shapes*.

This chapter describes a search procedure that finds shapes systematically. From a starting point $\mathbb{A}$, we take search steps, each of which adds information to $\mathbb{A}$. We stop when we have enough behavior of the regular (uncompromised) participants such that—with some activity of the adversary—we could have a full execution.

Each search step increases information in some way, i.e. it is an enrichment in the sense we have been using, and which we will formalize in the notion of *homomorphism*. Our examples so far have illustrated the three most important ways to add information. We can add information by adding listener nodes to witness for the assumption that a value is disclosed (Case 1 of Principles 1.1 and 1.3). We can add information by adding new protocol message receptions and transmissions that help to explain those that are already present (Case 2 of Principles 1.1 and 1.3). And we can add information by identifying different parameters, as we identified C and D with A and B respectively, to produce skeleton $\mathbb{B}_{21}$. When there are different possible ways to explain some one aspect of existing behavior, the search branches.

We have implemented this search in a tool called CPSA, a Cryptographic Protocol Shape Analyzer. The core purpose of this chapter is to explain the theory underlying CPSA. We will say very little about it as software, and we will take another occasion to describe how a (now very informative) piece of software was designed to implement this theory.

We prove two main results. They show that the search steps may be viewed as a process of refinement leading to the shapes:

1. The search steps are *sound* (Thm. 7.1), in the sense that—when we take a step—every possible execution compatible with the assumptions before the step is still compatible on at least one branch after the step.

2. The search steps are *complete* (Thm. 7.1), in the sense that every shape is reached by some finite sequence of search steps.

These results do not imply decidability for security goals, since in some cases the search enumerates an infinite set of shapes. One of these may be a counterexample to a goal, in which case we certainly learn that it is false. However, if a goal is in fact true, but $\mathbb{A}$ has infinitely many shapes, then we do not learn that it is true at any finite stage.

The shape search is related to Athena [15], which also searched for executions that extend a given partial description. Athena used a representation that included adversary behavior as well as regular strands. It also used a more straightforward backward search, in which the tool seeks all possible transmission points—whether adversary actions or regular strands—for each component of a reception node that cannot yet be explained. Athena later incorporated an early version of the authentication test method (Principles 1.1 and 1.3) to prune its search [13]. Cremers's Scyther [4, 3] refined Athena's ideas, combining them with the notion of *characterization* [7], which we describe below in Section 3.3.

A strength of Scyther is that, unlike our search, it can be made to terminate in all cases, providing a bounded-session analysis when the unbounded-session analysis proves too costly. Our work differs in its emphasis on the authentication test method, and in its systematic treatment of enrichment via the skeletons-and-homomorphisms theory of Section 3.

1.5 Structure of this Chapter

Principles 1.1 and 1.3 have a different character from Principles 1.2 and 1.4. The former determine the potential explanations for unexplained behavior, and they drive the form of the search steps. By contrast, the latter are general observations about skeletons, about enrichments or *homomorphisms*—as we will call them—between them, and about executions. We will examine the latter in Section 3, after introducing the basic strand space terminology in Section 2.

We will then (Section 4) introduce a second example, a modified form of the Trusted Computing Group's protocol for constructing certificates for Attestation Identity Keys [1]. This suggests a strengthened version of Principle 1.3, which is parallel with Principle 1.1 in form.

In Section 5, we state a combined version of the two principles, and show that they characterize when a skeleton is an execution (Theorem 5.5). This in turn suggests a search-oriented version, which allows us to characterize the process of adding information to discover shapes; we will describe the latter in Section 6. Finally, the key results about the search process appear in Section 7.

2 Messages, Strands, Protocols

We assume that protocols send and receive values from an algebra of messages A. We introduce this algebra in two steps. We give an algebra of basic values A_0 in Section 2.1, from which we freely generate an algebra of messages by the two operations of encryption and tupling (Section 2.2). We regard the algebra A_0 as a sample, introduced for the sake of definiteness. The results in the remainder of this chapter rely only Lemma 2.3, presented in Section 2.3. Any choice of A_0 such that the resulting A satisfies Lemma 2.3 is acceptable.

Sections 2.4 and 2.5 define *strands* and *protocols* respectively.

2.1 Algebra of Basic Values

An algebra of basic values A_0 is shown in Fig. 11. It is the disjoint union of infinite sets of *nonces*, *symmetric keys*, *asymmetric keys*, *names*, and *texts*. We also write bkeys for $\mathsf{skeys} \cup \mathsf{akeys}$. The operator $\mathsf{sk}(a)$ maps names to signature keys, which are asymmetric keys. The operator $\mathsf{pk}(a)$ maps names to public encryption keys, also asymmetric keys. The operator $\mathsf{ltk}(a, b)$ maps two names to a symmetric key intended to be shared by them. Nothing in this set up ensures that a key $\mathsf{ltk}(a, b)$, $\mathsf{sk}(a)$, or $\mathsf{pk}(a)$ is actually uncompromised, i.e. used only in accordance with some protocol to be chosen later. This terminology simply

$$\text{Pairwise disjoint, infinite sets:} \quad \mathsf{nonces}, \mathsf{skeys}, \mathsf{akeys}, \mathsf{names}, \mathsf{texts}$$
$$\mathsf{A}_0 = \mathsf{nonces} \cup \mathsf{skeys} \cup \mathsf{akeys} \cup \mathsf{names} \cup \mathsf{texts}$$
$$\mathsf{sk}\colon \mathsf{names} \to \mathsf{akeys} \qquad \mathsf{pk}\colon \mathsf{names} \to \mathsf{akeys} \qquad \mathsf{ltk}\colon \mathsf{names} \times \mathsf{names} \to \mathsf{skeys}$$
$$\mathsf{bkeys} = \mathsf{skeys} \cup \mathsf{akeys} \qquad \mathsf{inv}\colon \mathsf{bkeys} \to \mathsf{bkeys}$$
$$\mathsf{inv}(K) \text{ is written } K^{-1}$$
$$(K^{-1})^{-1} = K \qquad K^{-1} \neq K \text{ iff } K \in \mathsf{akeys}$$
$$\mathsf{sk}(\cdot), \mathsf{pk}(\cdot), \mathsf{ltk}(\cdot,\cdot) \text{ are injective}$$
$$\mathsf{sk}(\cdot), \mathsf{pk}(\cdot), \mathsf{pk}(\cdot)^{-1} \text{ have disjoint ranges}$$

Figure 11: Algebra of basic values A_0

associates different kinds of keys with principals. Similarly, even though a value $N \in \mathsf{nonce}$, there is no guarantee that in some execution it will be generated only once. Nonces are simply data values of some particular form. We will express assumptions about keys being uncompromised or nonces being freshly chosen using a terminology—"non-originating values" and "uniquely originating values" to be introduced in Section 2.4. Let

$$\mathsf{LT}_0 = \mathrm{rng}(\mathsf{pk}(\cdot)) \cup \mathrm{rng}(\mathsf{sk}(\cdot)) \cup \mathrm{rng}(\mathsf{ltk}(\cdot,\cdot)) \quad \text{and} \quad \mathsf{LT} = \mathsf{LT}_0 \cup (\mathsf{LT}_0)^{-1}$$

be the set of long term keys. For $K \in \mathsf{LT}$, let $\mathsf{owners}(K)$ and $\mathsf{owners}(K^{-1})$ be the unique $\{a\}$ or $\{a,b\}$ for which $K = \mathsf{sk}(a)$, $K = \mathsf{pk}(a)$, or $K = \mathsf{ltk}(a,b)$. The range of $\mathsf{owners}(K)$ covers all of names.

A homomorphism $\eta\colon \mathsf{A}_0 \to \mathsf{A}_0$ is a map that respect sorts, and acts homomorphically on $\mathsf{sk}(a)$, $\mathsf{pk}(a)$, $\mathsf{ltk}(a,b)$, and K^{-1}.

2.2 Message Algebra

We regard protocols as acting on messages chosen from some algebra of messages A (Fig. 12). We will assume that the members of A are freely generated by *tupling* and *encryption* starting from an algebra A_0 of *basic values*, and a disjoint infinite set X of untyped variable-like values we call *indeterminates*. Members of A are called *messages*. The encryption operator, written $\{\!|t_0|\!\}_{t_1}$, must definitely be a free operation, and t_0 is the plaintext and t_1 is the key. For tupling, we assume a set of "tags" TAG. When $tag \in \mathsf{TAG}$, $n \geq 0$, and $t_1, \ldots, t_n \in \mathsf{A}$,

$$X \text{ is infinite} \qquad X, \mathsf{TAG}, \mathsf{A}_0 \text{ are pairwise disjoint} \qquad nil \in \mathsf{TAG}$$
$$\mathsf{A} \text{ is freely generated from } X \cup \mathsf{A}_0 \text{ by:}$$
$$\mathsf{enc}\colon \mathsf{A} \times \mathsf{A} \to \mathsf{A} \qquad \mathsf{tuple}\colon \mathsf{TAG} \times \mathsf{A}^* \to \mathsf{A}$$
$$\text{which are written (resp.) as:} \qquad \{\!|t_0|\!\}_{t_1} \qquad tag\ t_1 \mathbin{\hat{}} \ldots \mathbin{\hat{}} t_n$$
$$nil\ t_0 \mathbin{\hat{}} \ldots \mathbin{\hat{}} t_n \text{ is written as } t_0 \mathbin{\hat{}} \ldots \mathbin{\hat{}} t_n$$

Figure 12: Algebra A, given Basic Values A_0, Indeterminates X

then the tagged tuple $tag\ t_1 \hat{} \ldots \hat{} t_n \in \mathsf{A}$. For a distinguished tag *nil*, we write $nil\ t_0 \hat{} \ldots \hat{} t_n$ as $t_0 \hat{} \ldots \hat{} t_n$ with no visible tag.

Without significantly changing the remainder of this chapter, tupling could also be replaced somewhat different operations, for instance by a free binary pairing operation, or by an associative concatenation, or even, with some care, by an associative commutative operation.

Homomorphisms on the Message Algebra. A homomorphism $\alpha = (\eta, \chi)\colon \mathsf{A} \to \mathsf{A}$ consists of a homomorphism η on A_0 and a function $\chi\colon X \to \mathsf{A}$. It is defined for all $t \in \mathsf{A}$ by the conditions:

$$\begin{array}{ll} \alpha(a) = \eta(a), & \text{if } a \in \mathsf{A}_0 \\ \alpha(x) = \chi(x), & \text{if } x \in X \end{array} \qquad \begin{array}{c} \alpha(\{\!|t_0|\!\}_{t_1}) = \{\!|\alpha(t_0)|\!\}_{\alpha(t_1)} \\ \alpha(tag\ t_1 \hat{} \ldots \hat{} t_n) = tag\ \alpha(t_1) \hat{} \ldots \hat{} \alpha(t_n) \end{array}$$

Thus, basic values serve as typed variables, replaceable only by other values of the same sort, while indeterminates x are untyped "blank slots." Tags remain constant under homomorphisms. The *parameters* $\mathsf{params}(t)$ of a message t are determined inductively:

$$\begin{array}{lll} \mathsf{params}(t) = \mathsf{owners}(t) & \textbf{if} & t \in \mathsf{LT}; \\ \mathsf{params}(K) = \{K, K^{-1}\} & \textbf{if} & K \in \mathsf{bkeys} \setminus \mathsf{LT}; \\ \mathsf{params}(a) = \{a\} & \textbf{if} & a \in \mathsf{A}_0 \text{ is not a basic key}; \\ \mathsf{params}(x) = \{x\} & \textbf{if} & x \in X; \\ \mathsf{params}(t) = \bigcup_i \mathsf{params}(t_i) & \textbf{if} & t = tag\ t_1 \hat{} \ldots \hat{} t_n \text{ or } t = \{\!|t_1|\!\}_{t_2}. \end{array}$$

Thus, the parameters of any message make up a finite set of basic values and indeterminates. Moreover, if α, β are homomorphisms that agree on $\mathsf{params}(t)$, then $\alpha(t) = \beta(t)$. Conversely, in this algebra, if α, β differ on any argument in $\mathsf{params}(t)$, then $\alpha(t) \neq \beta(t)$. However, in other relevant algebras this may no longer hold. Finally, the non-parameters are precisely the long-term keys. Writing

$$\mathsf{PARAMS} = \{v\colon \exists t,\ v \in \mathsf{params}(t)\},$$

we have, for all $a \in \mathsf{A}_0$, $a \in \mathsf{PARAMS}$ iff $a \notin \mathsf{LT}$.

Messages are abstract syntax trees in the usual way:

Definition 2.1 *1. Let ℓ and r be the partial functions where:*

$$\begin{array}{rll} t = \{\!|t_1|\!\}_{t_2} & \textit{implies} & \ell(t) = t_1 \textit{ and } r(t) = t_2; \\ t = tag\ t_1 \hat{} t_2 \hat{} \ldots \hat{} t_j & \textit{implies} & \ell(t) = t_1 \textit{ and } r(t) = t_2 \hat{} \ldots \hat{} t_j; \\ t \in \mathsf{A}_0 & \textit{implies} & \ell(t) \textit{ and } r(t) \textit{ are undefined.} \end{array}$$

2. A path *p is a sequence in $\{\ell, r\}^*$. We write $\mathsf{cons}(f, p)$ for the sequence whose first member is f and whose successive elements are those of p. We write $p_1 \frown p_2$ for the result of appending p_2 to the end of p_1.*

We regard p as a partial function, where $\langle\rangle = \mathsf{Id}$ and $\mathsf{cons}(f, p) = p \circ f$. When the rhs is defined, we have:

(a) $\langle\rangle(t) = t$;

(b) $\mathsf{cons}(\ell, p)(t) = p(\ell(t))$*; and*

(c) $\mathsf{cons}(r, p)(t) = p(r(t))$.

3. p encounters a key edge *in* t *if* $p_1(t)$ *is an encryption, where* $p = p_1 \frown \langle r \rangle \frown p_2$.

4. t_0 is an ingredient of t, *written* $t_0 \sqsubseteq t$, *if* $t_0 = p(t)$ *for some* p *that does not encounter a key edge in* t.

5. t_0 appears in t, *written* $t_0 \ll t$, *if* $t_0 = p(t)$ *for some* p.

6. p traverses a member of S *in* t *if* $p = p_1 \frown p_2$, *where* $p_1(t) \in S$ *and* $p_2 \neq \langle\rangle$.

As an example, consider the message $t = \{\!|\{\!|k|\!\}_{\mathsf{sk}(A)}|\!\}_{\mathsf{pk}(B)}$; for

$p_0 = \langle \ell, \ell \rangle$**,** we have $k = p_0(t)$. Since p_0 does not encounter a key edge, $k \sqsubseteq \{\!|\{\!|k|\!\}_{\mathsf{sk}(A)}|\!\}_{\mathsf{pk}(B)}$.

$p_1 = \langle r \rangle$**,** we have $\mathsf{pk}(B) = p_1(t)$. However, since p_1 encounters a key edge, we have established only the weaker $\mathsf{pk}(B) \ll \{\!|\{\!|k|\!\}_{\mathsf{sk}(A)}|\!\}_{\mathsf{pk}(B)}$.

$p_2 = \langle \ell, r \rangle$**,** we have $\mathsf{sk}(A) = p_2(t)$. Since p_1 again encounters a key edge, we have only $\mathsf{sk}(A) \ll \{\!|\{\!|k|\!\}_{\mathsf{sk}(A)}|\!\}_{\mathsf{pk}(B)}$.

In $\{\!|s|\!\}_k$, only the path $\langle r \rangle$ leads to k. Hence, $k \ll \{\!|s|\!\}_k$ but $k \not\sqsubseteq \{\!|s|\!\}_k$. Since $\langle \ell \rangle$ leads to s, $s \sqsubseteq \{\!|s|\!\}_k$.

2.3 Properties of Homomorphisms

This A and its finitely generated homomorphisms (which we will focus on henceforth) have a number of properties we will rely on.

Definition 2.2 1. *A parameter* $v \in \mathsf{PARAMS}$ *is a* source parameter *of a homomorphism* α *if* $\alpha(v) \neq v$; $w \in \mathsf{PARAMS}$ *is a* target parameter *if for some source parameter* v, $w \ll \alpha(v)$.

2. *A homomorphism* α *is* finitely generated *iff it has finitely many source parameters. In this case it has finitely many target parameters, also.*

3. *Messages* $v, w \in \mathsf{A}$ have a common instance *iff* $\alpha(v) = \alpha(w)$ *for some* α.

4. *A homomorphism* α *is an* isomorphism *on* A*, or a* renaming*, iff there exists a* β *such that* $\beta \circ \alpha$ *is the identity. We say that* α, α' *are* isomorphic *if for some renaming* ι, $\iota \circ \alpha = \alpha'$.

5. *If* α *and* γ *are finitely generated, and there exists a* β *such that* $\gamma = \beta \circ \alpha$*, then we say that* γ *is* at least as specific as α*, and write* $\alpha \leq_s \gamma$.

If ι is a finitely generated isomorphism, then we may regard it as a disjoint union of two functions. One is a permutation π of the source parameters of ι; π has no fixed points and maps finitely many basic values to basic values of the same sort, and finitely many indeterminates to indeterminates. The second is simply the identify function on the remaining basic values and ideterminates. Indeed, any function of this form is an isomorphism.

Hence, being isomorphic is an equivalence relation on finitely generated homomorphisms. It is reflexive and transitive because the identity is an isomorphism, and because isomorphisms are closed under composition. For symmetry, reason as follows. If α, β are finitely generated, and $\beta = \iota \circ \alpha$, then ι is finitely generated. Thus, ι is the disjoint union of a finite permutation π and an identity function. Thus, building ι' using π^{-1} and the same identity function, $\alpha = \iota' \circ \beta$.

A *preorder* means a reflexive, transitive relation.

Lemma 2.3 *1. $\leq_s$ is a preorder on finitely generated homomorphisms, and $\alpha \leq_s \gamma \leq_s \alpha$ implies that α and γ are isomorphic. Hence, $\leq_s$ is a partial order on isomorphism classes.*

2. When, for finitely generated α, β, γ, we have $\gamma = \beta \circ \alpha = \beta' \circ \alpha$, then $\beta(a) = \beta'(a)$ for all $a \in \mathrm{rng}(\alpha)$. Thus, the choice of β in Def. 2.2, Clause 5 is unique on $\mathrm{rng}(\alpha)$.

3. For any finitely generated γ, the set $\{\alpha \colon \alpha \leq_s \gamma\}$ contains only finitely many non-isomorphic members.

Lemma 2.4 *1. Assume $v, w \in \mathsf{A}$ have a common instance. There exists a finitely generated γ which is a most general unifier of v and w. That is, $\gamma(v) = \gamma(w)$, and whenever $\gamma'(v) = \gamma'(w)$, then $\gamma \leq_s \gamma'$.*

2. Let $v = \alpha(u)$ and $w = \beta(u)$, where α, β are finitely generated. Then α, β have a most specific common generalization γ for u. That is, there is a finitely generated γ such that:

(a) For some α_1, $v = (\alpha_1 \circ \gamma)(u)$,

(b) For some β_1, $w = (\beta_1 \circ \gamma)(u)$, and

(c) if γ' satisfies Clauses 2a–2b, then $\gamma' \leq_s \gamma$.

By Clause 1, there is a most general simultaneous unifier γ for the sequences $r_1, \ldots, r_k$ and $t_1, \ldots, t_k$ if any α simultaneously unifies each message of the first sequence with the corresponding message in the second. By Clause 3, $\leq_s$ (on finitely generated homomorphisms) is a well-founded partial order, to within isomorphism. *A fortiori*, $\{\beta \colon \alpha \leq_s \beta \wedge \beta \leq_s \gamma\}$ is finite for finitely generated γ.

Lemma 2.3 gives our central assumptions on the algebra. We believe that any algebra satisfying Lemma 2.3, and having a free encryption, and a tupling operator—or a pairing, concatenation, or multiset operation that is otherwise free—will satisfy the remaining results in this chapter.

With algebras where a finite set of unifiers cover the common instances of any two messages, in place of Clause 1, and a finite set of homomorphisms

replace the most specific common generalization of Clause 2, we believe that an easily adjusted version of our method is usable.

By contrast, in algebras with an operation like exclusive-or, which may be applied to tuples or encryptions rather than just basic values, and is subject to equational laws, other entirely new ideas are required.

2.4 Strands and Origination

A single local session of a protocol at a single principal is a *strand*, containing a linearly ordered sequence of transmissions and receptions that we call *nodes*. A transmission of message t is a directed term $+t$, and a reception of message t is a directed term $-t$.

We write $s \downarrow i$ for the i^{th} node on strand s, using 1-based indexing. We write $n \Rightarrow m$ when n, m are successive nodes on the same strand, i.e. when for some s, i, $n = s \downarrow i$ and $m = s \downarrow i + 1$. $\Rightarrow^+$ is the transitive closure of $\Rightarrow$. We write $\mathsf{msg}(n)$ for the message sent or received on the node n.

Origination. A message t_0 *originates* at a node n_1 if (1) n_1 is a transmission node; (2) $t_0 \sqsubseteq \mathsf{msg}(n_1)$; and (3) whenever $n_0 \Rightarrow^+ n_1$, $t_0 \not\sqsubseteq \mathsf{msg}(n_0)$. Thus, t_0 originates when it was transmitted without having been either received or transmitted (as an ingredient) previously on the same strand.

Values assumed to originate only on one node in an execution—*uniquely originating* values—formalize the idea of freshly chosen, unguessable values. Values assumed to originate nowhere may be used to encrypt or decrypt, but are never sent as message ingredients. They are called *non-originating* values. For a non-originating value K, $K \not\sqsubseteq t$ for any transmitted message t. However, $K \ll \{|t_0|\}_K \sqsubseteq t$ possibly, which is why we distinguish $\sqsubseteq$ from $\ll$.

As an example, our comment in applying Principle 1.3 to Fig. 2 that k is not retransmitted by any strand that has received it is meant in the sense of $\sqsubseteq$. We never have $k \sqsubseteq \mathsf{msg}(n)$ if n is a transmission node, and for some reception node m, $m \Rightarrow^+ n$ and $k \sqsubseteq \mathsf{msg}(m)$. However, we may have $k \ll \mathsf{msg}(n)$, which is harmless because it does not contribute to disclosure of k.

We say that n is *the origin of* t_0 in a set of nodes S if (1) $n \in S$, (2) t_0 originates at n, and (3) t_0 originates at no other node in S. It is *uniquely originating in* S if for some n, n is the origin of t_0 in S.

We say that t_0 is *non-originating in* S if there is no $n \in S$ such that $t_0 \sqsubseteq \mathsf{msg}(n)$. Evidently, if there is any $n \in S$ such that $t_0 \sqsubseteq \mathsf{msg}(n)$, then any full execution extending S will have to provide an origin for t_0.

Principals and Other Parameters. We extend the notion of *parameters* from messages to nodes and strands cumulatively. Suppose that s is a strand of length ℓ, and $i \leq \ell$; then

$$\mathsf{params}(s \downarrow i) = \bigcup_{0<j\leq i} \mathsf{params}(\mathsf{msg}(s \downarrow j)),$$

and $\mathsf{params}(s) = \mathsf{params}(s \downarrow \ell)$. Thus, the parameters to a node are those potentially varying arguments which can affect the messages that have been

sent or received up to and including that node. A strand s has only finitely many parameters, and any two homomorphisms that agree on those parameters will have the same action on messages sent and received along s.

In the model we use in this paper, there is no relevant entity acting as a principal. There are only names, and these names serve as parameters to some strands. Names are also associated with keys via the functions $\mathsf{ltk}(\cdot,\cdot)$, $\mathsf{pk}(\cdot)$, $\mathsf{sk}(\cdot)$. Thus, although informally we view several strands as being ongoing activities of a single principal at a particular time, in the model, there are only strands that share a particular name parameter and use the keys associated with that parameter. When a protocol may manipulate some long-term state belonging to the principals executing it, then we work in a richer model [10].

2.5 Protocols

A *protocol* Π is a finite set of strands, called the *roles* of Π, together with some constraints on unique and non-origination. We assume that every protocol Π contains the listener role $\mathsf{Lsn}[x]$, which consists of a single reception node $\stackrel{x}{\rightarrow} \bullet$. Instances of this listener role have already appeared in Figs. 2, 4, 6–7.

The constraints on origination for roles, which we will describe more specifically later, may be used to ensure that a particular role always contributes a uniquely originating value to an execution, as for instance a session key server may be trusted always to choose a fresh and unguessable session key. They may also be used to ensure that a particular role always involves a non-originating key. For instance, a protocol may ensure that the certificate authority role always uses a non-originating signing key. In modeling SEP, role origination constraints were not needed; we assumed only that particular values in a skeleton were non-originating or uniquely originating. There was no need to assume that every time we added a strand, some parameters to it would satisfy origination constraints.

Indeterminates represent syntactically unconstrained messages received from protocol peers, or passed down as parameters from higher-level protocols. Thus, we require an indeterminate to be received as an ingredient before appearing in any other way:

If n_1 is a node on $\rho \in \Pi$, with an indeterminate $x \ll \mathsf{msg}(n_1)$,

then $\exists n_0$, $n_0 \Rightarrow^* n_1$, where n_0 is a reception node and $x \sqsubseteq \mathsf{msg}(n_0)$.

As an example, the two strands shown in Fig. 1 are the roles of SEP. Our analysis of SEP did not rely on any origination constraints. Finally, it can be seen that if s were an indeterminate, then the responder role violates the principle that indeterminates are received before appearing in any other way. We instead interpret s as a basic value. This requirement is related to the requirement in constraint-solving approaches—originating with [12]—that variables in constraint sequences always appear first on the right-hand side of a constraint. The right hand sides represent message receptions, where the adversary must generate some instance of the constraint, and may select a value for the variable that makes this possible.

3 Skeletons, and Homomorphisms

A *skeleton* $\mathbb{A}$ consists of (possibly partially executed) regular strands, i.e. a finite set of nodes, $\mathsf{nodes}(\mathbb{A})$, with two additional kinds of information:

1. A partial ordering $\preceq_{\mathbb{A}}$ on $\mathsf{nodes}(\mathbb{A})$;
2. Finite sets $\mathsf{unique}_{\mathbb{A}}, \mathsf{non}_{\mathbb{A}}$ of basic values assumed uniquely originating and respectively non-originating in $\mathbb{A}$.

More formally:

Definition 3.1 *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; moreover, for all* $n_1 \in \mathsf{nodes}$*, if* $n_0 \Rightarrow^+ n_1$ *then* $n_0 \in \mathsf{nodes}$*;*
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 finite set of basic keys, and*
   (a) $\forall K \in \mathsf{non}, \forall n \in \mathsf{nodes}, K \not\sqsubseteq \mathsf{msg}(n)$*; and*
   (b) $\forall K \in \mathsf{non}, \exists n \in \mathsf{nodes}$*, either* $K \ll \mathsf{msg}(n)$ *or* $K^{-1} \ll \mathsf{msg}(n)$*;*
4. unique *is a finite set of basic values, and* $\forall a \in \mathsf{unique}, \exists n \in \mathsf{nodes}$ *s.t.* $a \sqsubseteq \mathsf{msg}(n)$*.*

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

5. *for all* $a \in \mathsf{unique}$*, if* a *originates at* $n_0 \in \mathsf{nodes}$*, then*
   (a) a *originates at no other node* $n_1 \in \mathsf{nodes}$*; and*
   (b) *if* $a \sqsubseteq \mathsf{msg}(n_1)$ *where* $n_1 \in \mathsf{nodes}$*, then* $n_0 \preceq n_1$*.*

The parameters *of* $\mathbb{A}$ *form the union:* $\mathsf{params}(\mathbb{A}) = \bigcup_{n \in \mathsf{nodes}(\mathbb{A})} \mathsf{params}(n)$*.*

All of the "skeletons" in Figs. 2–10 are skeletons, except that sometimes we have cheated on Clause 5b. Namely: In Fig. 2, the left hand node should precede the right hand node. In Fig. 6, the second skeleton $\mathbb{C}_{21}$ should have the node transmitting $\{|s|\}_k$ precede the listener node for s. In Fig. 7, the node transmitting t_0 should also precede the listener for k.

An object that violates Clause 3 cannot be enriched into one that satisfies it. By contrast, a preskeleton that violates Clause 5a or Clause 5b may sometimes be enriched to form a skeleton. In particular, it may be possible to map two nodes, at both of which some $a \in \mathsf{unique}$ originates, to a single node originating it, thereby satisfying Clause 5a. Likewise, we may be able to add edges to the ordering $\preceq$, while preserving its acyclicity, so as to satisfy Clause 5b. This is easily done for the non-skeletons in Figs. 2, 6, and 7.

3.1 Realized Skeletons

A skeleton is *realized* if its strands can really happen, independent of any other regular behavior. In a realized skeleton, the regular behavior present in the skeleton is combined with some *adversary* behavior, in a way that explains how each message received by a regular node could have been generated by that time. In particular, the skeleton's assumptions about non-origination and unique origination must not be violated by the adversary behavior.

Penetrator Webs. We represent the actions of the adversary by means of strands of certain special forms. These originate basic values or indeterminates; compose and transmit a tuple, having received its components; separate and transmit the components of a tuple, having received the tuple; encrypt and transmit a ciphertext, having received an encryption key and a plaintext; and decrypt and transmit a plaintext, having received the matching decryption key and a ciphertext:

Definition 3.2 *An* adversary strand *has any of the following forms:*

M_a: $\langle +a \rangle$ *where* a *is basic valueic* $\qquad$ M_g: $\langle +g \rangle$ *where* g *is an indeterminate*

C: $\langle -g \Rightarrow \ \cdots \Rightarrow \ -h \Rightarrow \ +tag\ g \mathbin{\hat{}} \ldots \mathbin{\hat{}} h \rangle$

S: $\langle -tag\ g \mathbin{\hat{}} \ldots \mathbin{\hat{}} h \Rightarrow \ +g \Rightarrow \ \cdots \Rightarrow \ +h \rangle$

E: $\langle -K \Rightarrow \ -h \Rightarrow \ +\{\!|h|\!\}_K \rangle$ $\qquad$ D: $\langle -K^{-1} \Rightarrow \ -\{\!|h|\!\}_K \Rightarrow \ +h \rangle$

Because an adversary uses a key only by means of E and D strands, it can use a key only if it receives that key value. Since every value that is received must have been originated, it follows that an adversary can never use a non-originating value. This justifies our use of "non-originating" to model "uncompromised."

The adversary can link together a number of strands, forming an "adversary web" that accomplishes some compound task.

Definition 3.3 (Adversary web, derivable) *Let* $G = \langle \mathcal{N}_G, (\rightarrow_G \cup \Rightarrow_G) \rangle$ *be a finite acyclic graph, where (i)* $n_0 \Rightarrow n_1$ *only if* n_0, n_1 *are successive nodes on an adversary strand; and (ii)* $n \rightarrow m$ *only if* n *is a transmission node,* m *is a reception node, and* $\mathsf{msg}(n) = \mathsf{msg}(m)$*.* G *is an* adversary web *with support* S_{spt} *and result* R *if* S_{spt} *and* R *are sets of messages and moreover:*

1. *If* $n_2 \in \mathcal{N}_G$ *is a reception node, then either* $\mathsf{msg}(n_2) \in S_{\mathrm{spt}}$ *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* $t \in S_{\mathrm{spt}}$ *or for some positive* $n \in \mathcal{N}_G$*,* $\mathsf{msg}(n) = t$*.*

If V *is a set of basic values, then term* t_1 *is* derivable from S_{spt} avoiding V *if there is a web* G *with support* $S_G \subseteq S_{\mathrm{spt}}$ *and* $t_1 \in R_G$*, where no basic value in* V *originates on a penetrator strand in* G*.*

This suggests the criterion for when a skeleton $\mathbb{A}$ represents a possible execution. $\mathbb{A}$ is a possible execution if, for each reception node n, there exists an adversary

web deriving $\mathsf{msg}(n)$ using messages transmitted earlier in $\mathbb{A}$, and avoiding basic values constrained by $\mathsf{non}_{\mathbb{A}}$ and $\mathsf{unique}_{\mathbb{A}}$. However, if $a \in \mathsf{unique}_{\mathbb{A}}$, but it does not originate on any node in $\mathbb{A}$, then it is in fact unconstrained: It can originate just once, but on an adversary M_a node, after which the adversary can use it as much as necessary. The constrained values in $\mathsf{unique}_{\mathbb{A}}$ are those that originate on a regular node of $\mathbb{A}$.

Definition 3.4 (Realized) *A basic value a is* to be avoided in $\mathbb{A}$ *if $a \in \mathsf{non}_{\mathbb{A}}$, or if $a \in \mathsf{unique}_{\mathbb{A}}$ and there exists $n \in \mathsf{nodes}_{\mathbb{A}}$ such that a originates on n.*

A reception node $m \in \mathsf{nodes}_{\mathbb{A}}$ is realized in $\mathbb{A}$ *if $\mathsf{msg}(m)$ is derivable from*

$$\{\mathsf{msg}(n) \colon n \prec_{\mathbb{A}} m \text{ and } n \text{ is a transmission node}\}$$

avoiding the set of basic values to be avoided in $\mathbb{A}$.

$\mathbb{A}$ is realized *if it is a skeleton and every reception node $m \in \mathsf{nodes}_{\mathbb{A}}$ is realized in $\mathbb{A}$.*

We have been saying informally that $\mathbb{A}$ represents a possible execution, and we now stipulate that this means that $\mathbb{A}$ is realized. The adversary webs explain how the adversary can generate values that will satisfy each reception node in $\mathbb{A}$.

Realized skeletons from Section 1 include $\mathbb{B}_{11}$ and $\mathbb{D}_{11}$. In the case of $\mathbb{B}_{11}$, the adversary web needed to "explain" its two reception nodes are each the empty web, since each reception node has a message that has already been transmitted on an earlier node. However, $\mathbb{D}_{11}$ requires an adversary web that decrypts $\{|\{|k|\}_{\mathsf{sk}(A)}|\}_{\mathsf{pk}(C)}$ using the decryption key $\mathsf{pk}(C)^{-1}$, and then re-encrypts with the public key $\mathsf{pk}(B)$, neither of which is to be avoided in $\mathbb{D}_{11}$. Skeleton $\mathbb{E}_{11}$ is again realized, using the empty adversary web.

By an old result [16, Lemma 2.9]:

Lemma 3.5 *If every node $m \in \mathsf{nodes}_{\mathbb{A}}$ is realized in the preskeleton $\mathbb{A}$, then Definition 3.1, Clause 5b is satisfied in $\mathbb{A}$.*

It is in fact easy to check whether there is an adversary web G that derives $\mathsf{msg}(m)$ from $\{\mathsf{msg}(n) \colon n \prec_{\mathbb{A}} m \text{ and } n \text{ is a transmission node}\}$. This is because we can always assume that the web is in *normal form* in the Prawitz-like sense [14] that a constructive adversary strand never precedes a destructive one [11].

3.2 Homomorphisms

The notion of enrichment used earlier will be formalized as a kind of homomorphism among skeletons, or more generally preskeletons. A (pre)skeleton homomorphism has two components. One is a homomorphism α on the underlying algebra A, which says how to transform the messages sent and received on nodes of the source (pre)skeleton to messages sent and received on the target (pre)skeleton. The other component is a map ϕ from nodes of the source

(pre)skeleton to nodes of the target (pre)skeleton. The most important condition is that $\mathsf{msg}(\phi(n)) = \alpha(\mathsf{msg}(n))$. That is, the message sent or received on the target node should be α applied to the message sent or received on the source node. A homomorphism's target may contain additional nodes not of the form $\phi(n)$.

Definition 3.6 *Suppose $\mathbb{A}, \mathbb{B}$ are preskeletons, α is a homomorphism on A, and $\phi\colon \mathsf{nodes}_{\mathbb{A}} \to \mathsf{nodes}_{\mathbb{B}}$. $H = [\phi, \alpha]$ is a* homomorphism *if*

1. *For all s, i, if $s \downarrow i \in \mathbb{A}$ then there is an s' s.t. for all $j \leq i$, $\phi(s \downarrow j) = s' \downarrow j$;*
2. *For all $n \in \mathbb{A}$, $\mathsf{msg}(\phi(n)) = \alpha(\mathsf{msg}(n))$;*
3. *n and $\phi(n)$ agree in direction, either transmission $+$ or reception $-$;*
4. *$n \preceq_{\mathbb{A}} m$ implies $\phi(n) \preceq_{\mathbb{B}} \phi(m)$;*
5. *$\alpha(\mathsf{non}_{\mathbb{A}}) \subseteq \mathsf{non}_{\mathbb{B}}$;*
6. *$\alpha(\mathsf{unique}_{\mathbb{A}}) \subseteq \mathsf{unique}_{\mathbb{B}}$;*
7. *If $a \in \mathsf{unique}_{\mathbb{A}}$ and a originates at $n \in \mathsf{nodes}_{\mathbb{A}}$, then $\alpha(a)$ originates at $\phi(n)$.*

We write $H\colon \mathbb{A} \mapsto \mathbb{B}$ when H is a homomorphism from $\mathbb{A}$ to $\mathbb{B}$. When α, α' agree on $\mathsf{params}(\mathbb{A})$, then $[\phi, \alpha] = [\phi, \alpha']$; i.e., $[\phi, \alpha]$ is the equivalence class of pairs under this relation.

We sometimes write ϕ_H and α_H to refer to a ϕ and α such that $H = [\phi, \alpha]$.

H is an inclusion map *if ϕ_H is the identity function. In this case, α_H is also the identity, i.e. $H = [\phi_H, \mathsf{Id}]$. $\mathbb{A}$ is a* subskeleton *of $\mathbb{B}$ if there is an inclusion $H\colon \mathbb{A} \mapsto \mathbb{B}$.*

We regard the finitely generated homomorphism α_0 which is the identity for all $v \notin \mathsf{params}(\mathbb{A})$ as the canonical representative of $[\phi, \alpha]$. Thus, henceforth, we will assume that the message homomorphism α in $[\phi, \alpha]$ is finitely generated.

The condition on origination in Clause 7 avoids the degeneracy in which a point of origination is destroyed for some basic value $a \in \mathsf{unique}_{\mathbb{A}}$. We stipulate that such degenerate maps are not homomorphisms.

Lemma 3.7 *If $\mathbb{A}$ is a preskeleton, then the identity $\mathsf{Id}_{\mathbb{A}}\colon \mathbb{A} \mapsto \mathbb{A}$. If $H\colon \mathbb{A} \mapsto \mathbb{B}$ and $J\colon \mathbb{B} \mapsto \mathbb{C}$, then $J \circ H\colon \mathbb{A} \mapsto \mathbb{C}$. Thus, skeletons and homomorphisms form a category.*

Section 1 incorporates many examples of homomorphisms, starting with an absence of homomorphisms. The deadness of Fig. 2 asserts that there are no homomorphisms from $\mathbb{A}_0$ whose target skeleton is realized. Figs. 3–4 illustrate a pair of homomorphisms $\mathbb{B} \mapsto \mathbb{B}_1$ and $\mathbb{B} \mapsto \mathbb{B}_2$. In each of these homomorphisms, $\mathbb{B}$ is included into a larger skeleton: the message homomorphism α is the identity and the node map ϕ is the identity in each case. The homomorphism $\mathbb{B}_1 \mapsto \mathbb{B}_{11}$

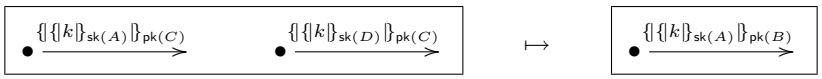


Figure 13: A non-injective homomorphism $\mathbb{A}_1 \mapsto \mathbb{A}_2$

is a bit more interesting, since its message homomorphism maps the names A, C to A and B, D to B. Its node map is surjective. By composition, we also have a homomorphism $\mathbb{B} \mapsto \mathbb{B}_{11}$.

The homomorphism $\mathbb{C} \mapsto \mathbb{C}_{21}$ is an embedding, with a second embedding $\mathbb{C}_{21} \mapsto \mathbb{C}_{211}$, and similar maps $\mathbb{D} \mapsto \mathbb{D}_1$ and $\mathbb{E} \mapsto \mathbb{E}_1$. We have not yet illustrated any case in which ϕ is non-injective. Fig. 13 is an artificial but simple example. A message homomorphism mapping A, D to the same value permits the two transmission nodes of $\mathbb{A}_1$ to be identified in the target $\mathbb{A}_2$. In this case, A, D are mapped to A, and C to B, but in fact the choice of target values is arbitrary. A renaming could change them to any other pair of distinct values, producing an isomorphic result.

In Fig. 13, there are also two homomorphisms H_1, H_2 in the opposite, right-to-left direction. One of them, say H_1, renames B to C and maps the single node of $\mathbb{A}_2$ to the left-hand node of $\mathbb{A}_1$. The other homomorphism H_2 renames B to C and also renames A to D; it maps the single node of $\mathbb{A}_2$ to the right-hand node of $\mathbb{A}_1$. These homomorphisms are, however, not essentially different. $\mathbb{A}_1$ has a non-trivial automorphism J, i.e. an isomorphism to itself. This is the map that interchanges the two nodes, while mapping A to D and D to A. In fact, $H_2 = J \circ H_1$, i.e. H_1 and H_2 differ by an isomorphism.

3.3 Describing Executions

An execution means a realized skeleton. We regard each skeleton $\mathbb{A}$ as describing a set of executions. This is the set of realized homomorphic images of $\mathbb{A}$.

A dead skeleton describes the empty set of executions. A realized skeleton describes a set of executions that includes itself. A skeleton that is neither dead nor realized describes a non-empty set that does not include itself.

Definition 3.8 *$\mathbb{A}$ is a* dead *skeleton if for every homomorphism $H\colon \mathbb{A} \mapsto \mathbb{B}$, $\mathbb{B}$ is not realized.*

Lemma 3.9 *If $\mathbb{A}$ is dead and $H\colon \mathbb{A} \mapsto \mathbb{B}$, then $\mathbb{B}$ is dead.*

Proof: If $J\colon \mathbb{B} \mapsto \mathbb{C}$, then $J \circ H\colon \mathbb{A} \mapsto \mathbb{C}$, so $\mathbb{C}$ is not realized. □

This lemma formalizes Principle 1.2.

Any non-dead skeleton $\mathbb{A}$ describes an infinite set of executions. Since $\mathbb{A}$ is non-dead, it has at least one homomorphism $H\colon \mathbb{A} \mapsto \mathbb{B}$ with $\mathbb{B}$ realized. Moreover, we can always take the disjoint union $2\mathbb{B} = \mathbb{B} \cup \mathbb{B}'$ of the realized skeleton $\mathbb{B}$ with a renaming $\mathbb{B}'$ of itself. If the renaming is chosen to use new

values that do not to overlap with the original ones, the disjoint union $2\mathbb{B} = \mathbb{B} \cup \mathbb{B}'$ will again be a realized skeleton. Clearly, we can again rename $2\mathbb{B}$ to $(2\mathbb{B})'$, taking a union to obtain $4\mathbb{B} = 2\mathbb{B} \cup (2\mathbb{B})'$. We can repeat this process *ad infinitum*.

Since the set of realized skeletons contains so many members, we would like to focus on compact but informative ways to characterize this set. If $\mathbb{A}$ is a skeleton, then a *characterization* for $\mathbb{A}$ is a set of homomorphisms $\mathcal{C}$ such that:

1. If $H \in \mathcal{C}$, then $H \colon \mathbb{A} \mapsto \mathbb{B}$ where $\mathbb{B}$ is realized; and
2. If $J \colon \mathbb{A} \mapsto \mathbb{C}$ where $\mathbb{C}$ is realized, then $J = K \circ H$ for some $H \in \mathcal{C}$ and some homomorphism K.

CPSA's goal is, given a "starting point" $\mathbb{A}$, to compute a characterization for $\mathbb{A}$.

However, a key choice was to decide which characterization to select. For instance, one would prefer an algorithm that produces *finite* characterizations rather than infinite ones when possible. However, there is another consideration. Fig. 13 illustrates that we may have a situation where $\mathbb{A}_1 \mapsto \mathbb{A}_2 \mapsto \mathbb{A}_1$; so should we then prefer the larger skeleton $\mathbb{A}_1$ or the smaller skeleton $\mathbb{A}_2$?

We have opted for the smaller skeleton $\mathbb{A}_2$. Why? Homomorphisms that are not surjective may not be interesting. For instance, recall the realized skeletons $\mathbb{A}, 2\mathbb{A}, 4\mathbb{A}, \ldots$ we mentioned immediately after Lemma 3.9. There are homomorphisms from $\mathbb{B}$ to each later member of the sequence, and they are not surjective, but the later members of the sequence do not tell us anything new. However, a non-injective homomorphism such as the one in Fig. 13 says something interesting, namely that two different events could be identified. For this reason, we define:

Definition 3.10 *1.* $H = [\phi, \alpha] \colon \mathbb{A} \mapsto \mathbb{B}$ *is* node-injective *iff* ϕ *is an injective function* $\phi \colon \mathsf{nodes}(\mathbb{A}) \to \mathsf{nodes}(\mathbb{B})$.

2. $H \leq_n J$ *iff for some node-injective* K, $J = K \circ H$.

3. A set $\mathcal{C}$ *is a* node-injective characterization *for* $\mathbb{A}$ *iff*

   (a) If $H \in \mathcal{C}$, *then* $H \colon \mathbb{A} \mapsto \mathbb{B}$ *where* $\mathbb{B}$ *is realized; and*

   (b) If $J \colon \mathbb{A} \mapsto \mathbb{C}$ *where* $\mathbb{C}$ *is realized, then* $H \leq_n J$ *for some* $H \in \mathcal{C}$.

   For brevity, we will write characterization *for node-injective characterizations from now on.*

4. $\mathcal{C} \leq_n \mathcal{C}'$ *iff for every* $H \in \mathcal{C}$, *there exists a* $J \in \mathcal{C}'$ *such that* $H \leq_n J$.

5. $\min(\mathcal{C}) = \{H \in \mathcal{C} \colon \forall J \in \mathcal{C}, J \leq_n H \text{ implies } H \leq_n J\}$.

CPSA's goal is in fact, given an $\mathbb{A}$, to compute the minimum characterization for $\mathbb{A}$. Here, the uniqueness in the words "the minimum" is meant to within isomorphism. We will write $\mathcal{C} \sim \mathcal{C}'$ iff every $H \in \mathcal{C}$ is isomorphic to some $H' \in \mathcal{C}'$ and vice versa. Then:

Lemma 3.11 *1. If $H\colon \mathbb{A} \mapsto \mathbb{A}$ is node-injective, then H is an isomorphism from $\mathbb{A}$ to itself.*

2. If $H \leq_n J \leq_n H$, then there is an isomorphism I such that $I \circ H = J$. Hence, $\leq_n$ is a partial order (to within isomorphism).

3. The relation $\leq_n$ is well founded (to within isomorphism).

4. $\{H\colon\ H\colon \mathbb{A} \mapsto \mathbb{B} \ \wedge\ \mathbb{B}$ is realized$\}$ is a characterization for $\mathbb{A}$.

5. $\min(\mathcal{C})$ is a non-empty characterization for $\mathbb{A}$ if $\mathcal{C}$ is.

6. If $\mathcal{C}, \mathcal{C}'$ are characterizations for $\mathbb{A}$, $\min(\mathcal{C}) \sim \min(\mathcal{C}')$.

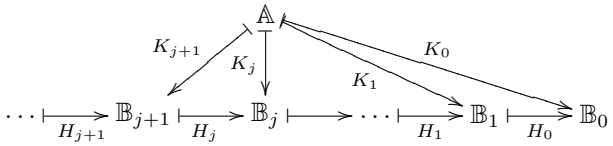
Proof:

1. If $H = [\phi, \alpha]\colon \mathbb{A} \mapsto \mathbb{A}$ is injective, then by the finiteness of $\mathsf{nodes}(\mathbb{A})$, ϕ is a bijection and does not introduce new pairs into the ordering. That is, $\phi(n) \preceq \phi(m)$ implies $n \preceq m$.

   So we must show next that the message algebra homomorphism α is invertible. However, by Lagrange's theorem, there is a k such that the k^{th} iterate ϕ^k of ϕ is the identity. If $k = 1$, then α is the identity. If $k = j+1$, then the j^{th} iterate α^j of α inverts α.

2. If $H \leq_n J \leq_n H$, then we have $J = G_1 \circ H$ and $H = G_2 \circ J$, where G_1, G_2 are node-injective. Hence $G_2 \circ G_1$ is a node-injective homomorphism from the target of H to itself. By Clause 1, $G_2 \circ G_1$ has an inverse F, i.e. $F \circ (G_2 \circ G_1)$ is the identity. Thus, by associativity of composition, $F \circ G_2$ is the desired inverse to G_1.

3. Let $\ldots K_{j+1} \leq_n K_j \leq_n \ldots K_0$ be an infinite descending chain of homomorphisms in the $\leq_n$ ordering, as in the following diagram, where each H_i is nodewise injective:



   We want to ensure that all but finitely many of the H_i are isomorphisms. By Lemma 2.3, Clause 3, all but finitely many of the α_{K_i} are isomorphic to each other, hence all but finitely many of the α_{H_i} are isomorphisms. For convenience, assume that all α_{H_i} with $i > N_0$ are the identity. Thus, all of the H_i for $i > N_0$ are inclusion maps. But by finiteness of skeletons, all but finitely many must be the identity.

4. Clause 3a of Def. 3.10 is immediate. Clause 3b holds because $J \leq_n J$.

5. Non-emptiness of $\min(\mathcal{C})$ follows from well-foundedness. Observe that if $H \in \mathcal{C} \setminus \min(\mathcal{C})$, then there is some $J \in \min(\mathcal{C})$ such that $J \leq_n H$. Since $\mathcal{C}$ is a characterization, for any $K \colon \mathbb{A} \mapsto \mathbb{C}$ with $\mathbb{C}$ realized, there is a $H \in \mathcal{C}$ such that $H \leq_n K$. If $H \notin \min(\mathcal{C})$, then there is some $J \in \min(\mathcal{C})$ with $J \leq_n H \leq_n K$, so $J \leq_n K$. Thus, $\min(\mathcal{C})$ is a characterization.

6. If either $\mathcal{C}$ or $\mathcal{C}'$ is empty, then so is the other (i.e. $\mathbb{A}$ is dead); in this case the result is immediate.

   Assume both non-empty. Since $\min(\mathcal{C})$ is a characterization, for every $J \in \min(\mathcal{C}')$ there is a $H \in \min(\mathcal{C})$ such that $H \leq_n J$. Since $\min(\mathcal{C}')$ is a characterization, there is also a $K \in \min(\mathcal{C}')$ such that $K \leq_n H \leq_n J$. By the definition of min, $J \leq_n K$. Hence H, J differ by an isomorphism. Symmetry of $\min(\mathcal{C}), \min(\mathcal{C}')$ completes the proof.

□

This establishes that the CPSA goal, to compute the minimum characterization, is well-defined.

Definition 3.12 *The* shapes *for a skeleton $\mathbb{A}$ are the members of the nodewise minimum characterization for $\mathbb{A}$.*

The shapes for any skeleton $\mathbb{A}$ form a well-defined set, since we may apply Lemma 3.11, Clauses 5–6 to the characterization $\mathcal{C}$ containing all homomorphisms from $\mathbb{A}$ to realized skeletons. If $\mathbb{A}$ is dead, this set is empty. We may now justify Principle 1.4 directly from the definitions.

Lemma 3.13 *Suppose $\mathbb{A}$ has shapes $\mathcal{C}$. Suppose $H \colon \mathbb{A} \mapsto \mathbb{B}$, and $J \colon \mathbb{B} \mapsto \mathbb{D}$ where $\mathbb{D}$ is realized. There is some $K \colon \mathbb{A} \mapsto \mathbb{S}$ with $K \in \mathcal{C}$, and some node-injective $L \colon \mathbb{S} \mapsto \mathbb{D}$ such that $J \circ H = L \circ K$.*

3.4 The Hull of a Preskeleton

Suppose $\mathbb{A}$ is a preskeleton but not a skeleton. Then there is some $a \in \mathsf{unique}_{\mathbb{A}}$ which either originates at two or more nodes (Def. 3.1, Clause 5a), or else a is an ingredient in some node that does not follow the origin of a. In this subsection, we describe how to "fix" those situations, when they can be fixed. There is a single, canonical, most general way to do so.

A map f is *universal* in some set of maps F if $f \in F$ and, for every $f' \in F$, there is exactly one g such that f' is of the form $f' = g \circ f$.

Lemma 3.14 *Suppose $\mathbb{A}, \mathbb{B}$ are preskeletons, with $H \colon \mathbb{A} \mapsto \mathbb{B}$.*

1. *If $\gamma \leq_s \alpha_H$, then there is a $\mathbb{B}_0$ and a $G \colon \mathbb{A} \mapsto \mathbb{B}_0$ such that G is universal among all homomorphisms K with source $\mathbb{A}$ where $\gamma \leq_s \alpha_K$.*

2. *Suppose that $\preceq_{\mathbb{A}} \subseteq \preceq_1$ and $\phi_H(\preceq_1) \subseteq \preceq_{\mathbb{B}}$. Then there is a $\mathbb{B}_0$ and a $G \colon \mathbb{A} \mapsto \mathbb{B}_0$ such that G is universal among all homomorphisms $K \colon \mathbb{A} \mapsto \mathbb{B}_1$ where $\phi_K(\preceq_1) \subseteq \preceq_{\mathbb{B}_1}$.*

3. *If* $\phi_H(n_0) = \phi_H(n_1)$ *for* $n_0, n_1 \in \mathsf{nodes}(\mathbb{A})$*, then there is a* $\mathbb{B}_0$ *and a* $G\colon \mathbb{A} \mapsto \mathbb{B}_0$ *such that* G *is universal among all homomorphisms from* $\mathbb{A}$ *which identify* n_0 *and* n_1*.*

Proof:

1. Define $\mathsf{nodes}(\mathbb{B}_0)$ by applying γ to each strand s that contributes to $\mathbb{A}$, and let ψ be the bijection that maps each node $s \downarrow i \in \mathsf{nodes}(\mathbb{A})$ to $\gamma(s) \downarrow i$. Let $\preceq_{\mathbb{B}_0} = \psi(\preceq_{\mathbb{A}})$, $\mathsf{non}_{\mathbb{B}_0} = \gamma(\mathsf{non}_{\mathbb{A}})$, and $\mathsf{unique}_{\mathbb{B}_0} = \gamma(\mathsf{unique}_{\mathbb{A}})$.

   Then $\mathbb{B}_0$ is a preskeleton unless Def. 3.1, Clause 3a fails. However, if $a \sqsubseteq \mathsf{msg}(n) \in \mathsf{nodes}(\mathbb{B}_0)$ but $a \in \gamma(\mathsf{non}_{\mathbb{A}})$, then this property is preserved under composition. Thus, since $\gamma \leq_s \alpha_H$, and $\mathbb{B}$ satisfies Clause 3a, so does $\mathbb{B}_0$.

   Moreover, $[\psi, \gamma]$ is a homomorphism unless Def. 3.6, Clause 7 fails. However, if a originates on $s \downarrow i$, but $\gamma(a) \sqsubseteq \mathsf{msg}(\psi(s \downarrow j))$ for $j < i$, then $\alpha(a)_H \sqsubseteq \mathsf{msg}(\phi_H(s \downarrow j))$, contradicting the assumption that H is a homomorphism.

2. Define $\mathbb{B}_0$ to be the same as $\mathbb{A}$, except that the ordering is $\preceq_1$. This ordering is acyclic because its image under ϕ_H is acyclic.

3. We may suppose that $n_0 = s_0 \downarrow i$ and $n_1 = s_1 \downarrow i$, since if the two nodes lie at different indices, no homomorphism can identify them. Then the messages $\mathsf{msg}(s_0 \downarrow 1), \ldots, \mathsf{msg}(s_0 \downarrow i)$ are simultaneously unifiable with $\mathsf{msg}(s_1 \downarrow 1), \ldots, \mathsf{msg}(s_1 \downarrow i)$, since α_H equates them. Let γ be their simultaneous m.g.u. Apply Clause 1 to this γ, obtaining $G_0 = [\psi_0, \gamma]\colon \mathbb{A} \mapsto \mathbb{B}_0$.

   By Clause 2, we may extend the ordering $\preceq_{\mathbb{B}_0}$ so that $s_0 \downarrow j$ precedes (succeeds) every node that $s_1 \downarrow j$ precedes (succeeds), and vice versa.

   We now construct $\mathbb{B}_1$ by selecting the strand $\gamma(s_0)$. Let ψ_1 extend ψ_0 by mapping the nodes $s_1 \downarrow j$ to $s_0 \downarrow j$, discarding the unnecessary nodes. $G_1 = [\psi_1, \gamma]$ is the desired homomorphism.

□

Lemma 3.14 is used in the next proof. However, it is also used repeatedly in the proofs of Section 7.

Lemma 3.15 (Hull) *Let* $\mathbb{A}$ *be a preskeleton with* $H\colon \mathbb{A} \mapsto \mathbb{B}$*. If* $\mathbb{B}$ *is a skeleton, then there is a skeleton* $\mathbb{B}_0$*, and a* $G_{\mathbb{A}}\colon \mathbb{A} \mapsto \mathbb{B}_0$*, such that* $G_{\mathbb{A}}$ *is universal among all homomorphisms from* $\mathbb{A}$ *to skeletons.*

Proof: If $\mathbb{A}$ is a preskeleton but not a skeleton, then there is a counterexample either to Def. 3.1, Clause 5a or else to Def. 3.1, Clause 5b. In the first case, there are two nodes n_0, n_1 at both of which the same $a \in \mathsf{unique}_{\mathbb{A}}$ originates. By Def. 3.6, Clause 7, $\alpha_H(a)$ originates at $\phi_H(n_0)$ and at $\phi_H(n_1)$. Since $\mathbb{B}$ is a skeleton, $\phi_H(n_0) = \phi_H(n_1)$. Thus, we may apply Lemma 3.14, Clause 3.

In the second case, for some $a \in \mathsf{unique}_{\mathbb{A}}$, $a \sqsubseteq \mathsf{msg}(n_1)$ but with the origin n_0 of a, $n_0 \not\preceq_{\mathbb{A}} n_1$. In this case, we apply Lemma 3.14, Clause 2.

If the result of a step is not a skeleton, we iterate; however, we must terminate: At each step of the first kind, we reduce the number of nodes. At each step of the second kind, we reduce the number of incomparable nodes. □

Definition 3.16 *The* hull *of* $\mathbb{A}$*, written* $\mathsf{hull}(\mathbb{A})$*, is the universal map* $G_{\mathbb{A}}$ *given in Lemma. 3.15, when it exists.*

We write $\mathsf{hull}_{\alpha}(\cdot)$ *for the partial map that carries any skeleton* $\mathbb{A}$ *to the homomorphism* $\mathsf{hull}(\alpha(\mathbb{A}))$*.*

We sometimes use the word *hull* to refer also to the target $\mathbb{B}_0$ of $G_{\mathbb{A}}$.[2]

4 Attestation Identity Protocol

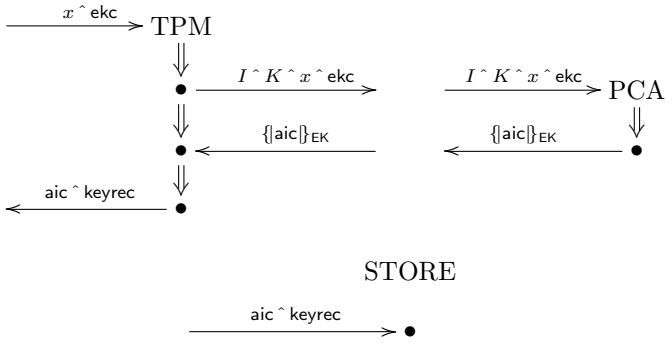
In Section 1, we examined SEP to extract a number of search principles. Of these, some concerned the structure of the search, which we have now formalized in Section 3. The two remaining principles—Principles 1.1 and 1.3—concern the individual steps in the search process. Unfortunately, however, Principle 1.3 is not yet strong enough to be complete. As formulated, it does not cover all the transformations that protocols apply to encrypted units. Instead, it covers only the most fundamental transformation, the act of creating the encrypted unit in the first place.

In this section, we will examine a second example, the Trusted Computing Group's protocol for generating certificates for "Attestation Identity Keys" (AIKs) [1]. These signature keys are intended to be resident within a Trusted Platform Module (TPM), and never to leave that device. However, the intention is that the certificate for an AIK public signature verification key K ensures that the private signature part K^{-1} is resident in *some* TPM, without allowing the recipient to determine which one. They provide, thus, anonymous assurance that signatures were prepared within some TPM.

The party that prepares certificates on AIKs is called a *privacy certificate authority*. It will prepare a certificate for any key K presented in a well-formatted message. So how does it ensure that the private part K^{-1} is TPM-resident? It encrypts the certificate aic using a public encryption key EK. That key is accompanied by a certificate from the TPM's manufacturer saying that the matching decryption key EK^{-1} is itself a long-term TPM-resident value. The TPM is designed to liberate the AIK certificate from this encryption only if the TPM holds the signature key that matches the verification key in the certificate.

The protocol itself is shown in Fig. 14 in slightly modified form. We will associate some non-origination assumptions with the PCA role in this protocol. First, when the PCA accepts an endorsement key certificate $\{\!|\, ekc\ \mathsf{MF} \,\hat{}\, \mathsf{EK} \,|\!\}_{\mathsf{sk}(\mathsf{MF})}$, it must check that the signing key is known to be the signature key of a recognized manufacturer. We model this by adding $\mathsf{sk}(\mathsf{MF})$ to the keys assumed to be

[2]The hull idea is due to Javier Thayer, as was the first proof of Lemma 3.15.



$$\mathsf{ekc} = \{\!|ekc\ \mathsf{MF}\hat{}\mathsf{EK}|\!\}_{\mathsf{sk(MF)}} \qquad \mathsf{aic} = \{\!|aic\ I\hat{}K\hat{}x|\!\}_{\mathsf{sk(PCA)}}$$
$$\mathsf{keyrec} = \{\!|aikrec\ K, K^{-1}|\!\}_{\mathsf{SRK}}$$

Figure 14: Modified Anonymous Identity Protocol MAIP

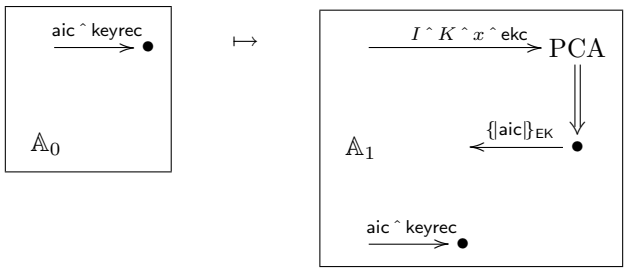
non-originating. Second, the point of the ekc is to vouch that the private part EK^{-1} is TPM-resident, and therefore used only in accordance with the rules. Hence, we also add EK^{-1} to the keys assumed to be non-originating.

In MAIP, there are two central transformations. The job of constructing and emitting an aic is one "transformation," which can be performed only by the privacy certifying authority. However, it is equally essential to the working of the protocol, that the PCA emits the aic only encrypted, and in such a way that the aic can be decrypted and transmitted in usable form only by a genuine TPM.

Of these two transformations, the first is certainly an instance of Principle 1.3. The value $\{\!|aic\ I\hat{}K\hat{}x|\!\}_{\mathsf{sk(PCA)}}$ is emitted, without having been contained as an ingredient of any previous node. Thus, if we assume that the signature key of PCA is uncompromised, any execution containing an instance of the STORE role must also contain an matching instance of the PCA role, as shown in Fig. 15. Observe that in $\mathbb{A}_1$ we have added $\mathsf{sk(MF)}, \mathsf{EK}^{-1}$ to the keys assumed non-originating, in accord with the origination constraint we associated with the TPM role.

The TPM's transformation to free the aic from its encryption is not an instance of Principle 1.3. The digitally signed unit must be received before being retransmitted. Thus, Principle 1.3, Clause 2 cannot apply. Moreover, Principle 1.1 does not apply. The AIK K may be a freshly chosen value, but it has already been transmitted outside all encryptions at the time that the PCA receives it. So Principle 1.1 implies nothing.

What we need here is an analog to Principle 1.1, but applying to encryptions rather than to fresh values. It needs one additional case, to cover the possibility that the adversary could independently generate the encryption. Thus, it would take the form:



$\mathsf{non}(\mathbb{A}_0) = \{\mathsf{sk}(\mathsf{PCA})\}$ $\mathsf{non}(\mathbb{A}_1) = \{\mathsf{sk}(\mathsf{PCA}), \mathsf{sk}(\mathsf{MF}), \mathsf{EK}^{-1}\}$

Figure 15: PCA Analysis, step 1 (Point of view: Store)

Principle 4.1 (The Encryption Test) *Suppose that $e = \{|t|\}_K$, is an encryption, and e is found in some message received in a skeleton $\mathbb{A}$ at a node n_1. Moreover, suppose that, in the message of n_1, e is found outside all of a number of encrypted forms $\{|t_1|\}_{K_1}, \ldots, \{|t_j|\}_{K_j}$. Then in any enrichment $\mathbb{B}$ of $\mathbb{A}$ such that $\mathbb{B}$ is a possible execution, either:*

1. *One of the matching decryption keys ${K_i}^{-1}$ is disclosed before n_1 occurs, so that e could be extracted by the adversary; or else*
2. *The encryption key K is disclosed before n_1 occurs, so that the adversary could construct $e = \{|t|\}_K$ from t; or else*
3. *Some regular strand contains a node m_1 in which e is transmitted outside the forms $\{|t_1|\}_{K_1}, \ldots, \{|t_j|\}_{K_j}$, but in all previous nodes $m_0 \Rightarrow^+ m_1$, e was found (if at all) only within the plaintexts $t_1 \ldots t_j$. Moreover, m_1 occurs before n_1.*

We may apply this principle to the encryption $e = \{|aic\ I \hat{} K \hat{} x|\}_{\mathsf{sk(PCA)}}$, with the single encrypted form $\{|\mathsf{aic}|\}_{\mathsf{EK}}$. If we assume that the signature key $\mathsf{sk}(\mathsf{PCA})$ is uncompromised, as well as the TPM-resident value EK^{-1}, then the first two disjuncts are inapplicable, and we are left with the regular TPM strand that transforms the aic from the form $\{|\mathsf{aic}|\}_{\mathsf{EK}}$ to aic.

We may observe, finally, that Principle 1.3 is a special case of Principle 4.1. If $j = 0$ in the list of encrypted forms $\{|t_1|\}_{K_1}, \ldots, \{|t_j|\}_{K_j}$—so that this is the empty list—then the first disjunct is unsatisfiable. Moreover, in the last disjunct, no earlier occurrences of e are permitted. Hence, the old principle is nothing but the $j = 0$ case.

Indeed, now Principles 1.1 and 4.1 are in essentially the same form. The only differences are that (1) the "critical ingredient" is a uniquely originating basic value c in Principle 1.1 and an encryption $e = \{|t|\}_K$ in Principle 4.1, and (2) the possibility that K becomes compromised is relevant only in Principle 4.1.

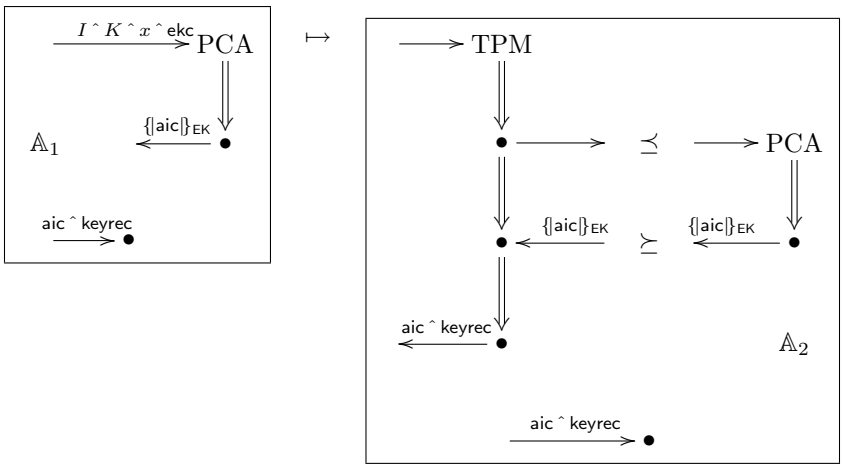


Figure 16: PCA Analysis, step 2 (Point of view: Store)

5 The Authentication Tests

We regard Principles 1.1 and 4.1 as specifying how certain *tests* can be solved. In each one, the critical value c or e is found only inside a number of encryptions $S = \{\{|t_1|\}_{K_1}, \ldots, \{|t_j|\}_{K_j}\}$, and is subsequently received at node n_1 outside of these forms S. The test is to explain how it is extracted from S. We call S the *escape set*, since the critical value does escape from it; indeed, it has done so before being received at n_1.

The solutions are of two kinds: Either a key is compromised, so the *adversary* can create an occurrence of c outside S, or else a *regular strand* has a transmission node m_1 which transmits c or e outside S, although earlier nodes on the same strand contained the critical value only within S (if at all). Since there are only finitely many roles in Π, unification on their nodes can find all candidates for regular solution nodes m_1. We formalize "being contained within S" as follows:

Definition 5.1 *Let S be a set of encryptions. A message t_0* is found only within *S in t_1, written $t_0 \odot^S t_1$, iff for every path p such that $p(t_1) = t_0$, either (1) p traverses a key edge or else (2) p traverses a member of S before its end.*

Message t_0 is found outside *S in t_1, written $t_0 \dagger^S t_1$, iff not $(t_0 \odot^S t_1)$.*

Equivalently, $t_0 \dagger^S t_1$ iff for some path p, (1) $p(t_1) = t_0$, (2) p traverses no key edge, and (3) p traverses no $e \in S$ before its end. Thus, $t_0 \sqsubseteq t_1$ iff $t_0 \dagger^{\emptyset} t_1$.

For instance, let $t_0 = \{\{|k|\}_{\mathsf{sk}(A)}|\}_{\mathsf{pk}(B)}$, and let $S_0 = \{t_0\}$ and $S_1 = \{\{|k|\}_{\mathsf{sk}(A)}\}$; because the sole path $\langle \ell, \ell \rangle$ to k in t_0 traverses first t_0 and then $\{|k|\}_{\mathsf{sk}(A)}$,

$$k \odot^{S_0} t_0 \quad \text{and} \quad k \odot^{S_1} t_0.$$

We used S_0 and S_1, respectively, in Section 1.1, when we considered A's point of view in SEP, and in Section 1.2, considering B's point of view. Moreover, for every S, $k \odot^S \{\!|s|\!\}_k$, because the only path to k traverses a key edge. However, $\{\!|k|\!\}_{\mathsf{sk}(A)} \dagger^{\emptyset} t_0$.

Taking key examples from MAIP next, we have

$$\mathsf{aic} \dagger^{\emptyset} \mathsf{aic} \quad \text{and} \quad \mathsf{aic} \dagger^{\emptyset} \{\!|\mathsf{aic}|\!\}_{\mathsf{EK}} \quad \text{but} \quad \mathsf{aic} \odot^{S_2} \{\!|\mathsf{aic}|\!\}_{\mathsf{EK}}$$

where $S_2 = \{\{\!|\mathsf{aic}|\!\}_{\mathsf{EK}}\}$. We formalize tests using *cuts*:

Definition 5.2 $\mathsf{Cut}(c, S, \mathbb{A})$*, the* test cut for c, S *in* $\mathbb{A}$*, is defined if* c *is a basic value or an encryption,* S *is a set of encryptions, and* $\exists n_1 \in \mathsf{nodes}(\mathbb{A})$ *such that* $c \dagger^S \mathsf{msg}(n_1)$*. In this case,*

$$\mathsf{Cut}(c, S, \mathbb{A}) = \{n \in \mathsf{nodes}(\mathbb{A}) \colon \exists m,\, m \preceq_{\mathbb{A}} n \wedge c \dagger^S \mathsf{msg}(m)\}.$$

Thus, in Fig. 2, again letting $\{\{\!|\{\!|k|\!\}_{\mathsf{sk}(A)}|\!\}_{\mathsf{pk}(B)}\} = S_0$,

$$\mathsf{Cut}(k, S_0, \mathbb{A}_0) = \{\, \bullet \overset{k}{\leftarrow} \,\},$$

i.e. the listener node at the right. In Fig. 3, the cut consists of the lower node:

$$\mathsf{Cut}(\{\!|s|\!\}_k, \emptyset, \mathbb{B}) = \{\, \bullet \overset{\{\!|s|\!\}_k}{\leftarrow} \,\}.$$

In Fig. 4, for both skeletons $\mathbb{B}_1$ and $\mathbb{B}_2$, we were interested in the test $\mathsf{Cut}(k, S_0, \mathbb{B}_i)$. In Figs. 6–7, it is $\mathsf{Cut}(s, S_3, \mathbb{C}_i)$, with $S_3 = \{\{\!|s|\!\}_k\}$. The cuts driving the MAIP analysis, shown in Figs. 15–16, are (with $S_2 = \{\{\!|\mathsf{aic}|\!\}_{\mathsf{EK}}\}$):

$$\mathsf{Cut}(\mathsf{aic}, \emptyset, \mathbb{A}_0) \quad \text{and} \quad \mathsf{Cut}(\mathsf{aic}, S_2, \mathbb{A}_0).$$

Definition 5.3 $Q = \mathsf{Cut}(c, S, \mathbb{A})$ *is* solved *if for every* $\preceq_{\mathbb{A}}$*-minimal* $m_1 \in Q$*:*

1. *either* m_1 *is a transmission node;*
2. *or there is a listener node* $m = \mathsf{Lsn}[K]$ *with* $m \prec_{\mathbb{A}} m_1$*, and either*
   (a) $c = \{\!|t_0|\!\}_K$*, or else*
   (b) *for some* $\{\!|t_0|\!\}_{t_1} \in S$*,* $K = t_1^{-1}$ *is the corresponding decryption key.*

The cut $\mathsf{Cut}(c, S, \mathbb{A})$ *is a* test *if it is unsolved. A* solution *to a cut is a transmission or listener node satisfying the clauses above.*

In skeleton $\mathbb{A}_0$ there is no way to add a solution to $\mathsf{Cut}(k, S_0, \mathbb{A}_0)$, and this showed that $\mathbb{A}_0$ is dead. Or to be more precise, in any homomorphic image of $\mathbb{A}_0$, the image of $\mathsf{Cut}(k, S_0, \mathbb{A}_0)$ remains unsolved.

The solution to $\mathsf{Cut}(k, S_0, \mathbb{B}_1)$ is an instance of Clause 2a, while the solution to $\mathsf{Cut}(k, S_0, \mathbb{B}_2)$, it is an instance of Clause 1.

The solutions to the MAIP cuts $\mathsf{Cut}(\mathsf{aic}, \emptyset, \mathbb{A}_0)$ and $\mathsf{Cut}(\mathsf{aic}, S_2, \mathbb{A}_0)$ are both instances of Clause 1; these solutions are added to form $\mathbb{A}_1$ and $\mathbb{A}_2$, resp.

Adversary webs (Def. 3.3) can derive messages iff the cuts are solved:

Lemma 5.4 *Let n be a reception node in $\mathbb{A}$. The following are equivalent:*

1. *There exists no adversary web G deriving $\mathsf{msg}(n)$ from*

$$\{\mathsf{msg}(m) : m \text{ is a transmission node } \wedge\, m \prec_{\mathbb{A}} n\}$$

   avoiding the set $\mathsf{non}_{\mathbb{A}} \cup (\mathsf{unique}_{\mathbb{A}} \cap \{a : a \text{ originates at some } n \in \mathbb{A}\})$.

2. *Node n is $\preceq_{\mathbb{A}}$-minimal in some well-defined, unsolved $\mathsf{Cut}(c, S, \mathbb{A})$.*

Proof: By [5, Props. 2–4]. □

Theorem 5.5 (Authentication Test Principle) *1. If every cut in $\mathbb{A}$ that is well-defined is solved, $\mathbb{A}$ is realized.*

2. If $\mathbb{A}$ is realized, then $\mathbb{A}$ has an extension $\mathbb{A}'$, obtained by adding only listener nodes, in which every well-defined cut is solved.

Proof: 1. Immediate from Lemma 5.4, Clause 1 implies Clause 2.
2. Since $\mathbb{A}$ is realized, for each reception node $n \in \mathsf{nodes}(\mathbb{A})$, there is an adversary web G deriving $\mathsf{msg}(n)$ from preceding transmission nodes. Build $\mathbb{A}'$ by adding—for each message t used as a key on an encryption or decryption strand in G—a listener node ℓ for t, where $\ell \prec_{\mathbb{A}'} n$ and (for all m) $m \prec_{\mathbb{A}'} n$ implies $m \prec_{\mathbb{A}'} \ell$. By construction, all of these listeners are derivable, since the adversary has in fact derived them in G.

Now apply Lemma 5.4, Clause 2 implies Clause 1 to $\mathbb{A}'$. □

6 Steps in the Search

Theorem 5.5, the Authentication Test Principle, describes what must be true at the end of the search, when we have reached a realized skeleton, namely that all the cuts are solved (modulo adding listener nodes). In this section, we "read off" what must happen step by step during the CPSA search. At each moment we are considering some skeleton $\mathbb{A}$, and Theorem 5.5 tells us that we must take a step if $\mathbb{A}$ has some well-defined but unsolved cut $\mathsf{Cut}(c, S, \mathbb{A})$. Thus, each step is taken in response to an unsolved cut.

There are three possible types of steps.

1. We may destroy the test $\mathsf{Cut}(c, S, \mathbb{A})$, rather than solving it.

   The step from $\mathbb{B}_2$ to $\mathbb{B}_{21}$ in Figs. 4–5 is an example. In Fig. 4 the cut $\mathsf{Cut}(k, \{\{|k|\}_{\mathsf{sk}(A)}\}, \mathbb{B}_2)$ is well-defined, and its minimum node is the upper right node, i.e. D's first step that receives $\{|\{|k|\}_{\mathsf{sk}(C)}|\}_{\mathsf{pk}(D)}$. The session key k is previously found only within $\{\{|k|\}_{\mathsf{sk}(A)}\}$. By applying the α that maps $[C \mapsto A, D \mapsto B]$, we destroy this cut; α leaves k and t_0 unchanged, but there is no node in which k is found outside of $\{t_0\}$ in $\mathbb{B}_{21}$.

   There is a second way to destroy a test. Suppose that a reception node n_1 is minimal in $\mathsf{Cut}(c, S, \mathbb{A})$. If there is a transmission node m_1 in $\mathbb{A}$ such

that $c \dagger^S \mathsf{msg}(m_1)$, but $n_1 \not\Rightarrow_{\mathbb{A}} m_1$, then we may be able to add the pair (m_1, n_1) to the ordering to destroy the test.

In the first case, the cut no longer needs to be solved, because it is no longer well-defined. In the second case, it no longer needs to be solved because n_1 is now no longer minimal in the cut.

2. We may add a regular transmission node m_1 that will satisfy Clause 1 of Def. 5.3. It must satisfy $c \dagger^S \mathsf{msg}(m_1)$, but it must be minimal in the cut, so in particular if $m_0 \Rightarrow^+ m_1$, then $c \odot^S \mathsf{msg}(m_0)$.

   The step from $\mathbb{B}$ to $\mathbb{B}_2$ in Figs. 3–4 is of this kind. So is the step from $\mathbb{D}$ to $\mathbb{D}_1$ in Fig. 8. Both steps in our analysis of MAIP are of this kind.

   The *absence* of steps is also important. Since in SEP $k \odot^S \{|s|\}_k$, for every S, the transmissions provided by the protocol do not free k from t_0. This helps to ensure the deadness of $\mathbb{A}_0$ in Fig. 2.

3. We may add a listener strand that will satisfy Clause 2 of Def. 5.3. It may add either a listener strand for a key that is the decryption key K^{-1} for some $\{|t|\}_K \in S$, or else the encryption key K when $c = \{|t|\}_K$.

   The step from $\mathbb{B}$ to $\mathbb{B}_1$ is of this kind (Figs. 3–4).

   In some cases the key K may be assumed non-originating in the skeleton, and then the step is impossible, since the resulting object would not satisfy the definition for a preskeleton (Def. 3.1, Clause 3a).

6.1 Sketch of Search Algorithm

How do we search for shapes, starting with a skeleton $\mathbb{A}_0$? At any point in the search, we have a set $\mathcal{F}$ of homomorphisms—all with source $\mathbb{A}_0$—that have been encountered, but not yet explored. Initially, we have the identity homomorphism of $\mathbb{A}_0$: $\mathcal{F} = \{\mathsf{Id}_{\mathbb{A}_0}\}$. We also have a set $\mathcal{S}$ of candidate shapes, initially empty.

If $\mathcal{F} = \emptyset$, then we are done, and return $\mathcal{S}$. Otherwise, select and remove any $H \colon \mathbb{A}_0 \mapsto \mathbb{A}_1 \in \mathcal{F}$.

If $\mathbb{A}_1$ is realized, and $J \not\leq_n H$ for any $J \in \mathcal{S}$, then we add H to $\mathcal{S}$. We discard any $J \in \mathcal{S}$ for which $H \leq_n J$. If $\mathbb{A}_1$ is not realized, then it has at least one unsolved test cut. If it has several, we may choose any one, since whichever unsolved test we choose, it must be solved or destroyed. If this cut has no solution, then $\mathbb{A}_1$ is dead, and we discard H. If this cut has a number of potential solutions, we construct a finite number of maximally general steps for that cut, placing them into $\mathcal{F}$.

We now explain how to select a test (Section 6.2), and how to select maximally general steps to solve that test (Sections 6.3–6.6).

6.2 Selecting a Test

If $\mathbb{A}_1$ is not realized, then it has unrealized reception nodes; let n_1 be any one of them. There is an unsolved cut $\mathsf{Cut}(c, S, \mathbb{A}_1)$ of which n_1 is a minimal member.

Since $c \dagger^S \mathsf{msg}(n_1)$, $c \sqsubseteq \mathsf{msg}(n_1)$. We consider each of the basic values and encryptions c such that $c \sqsubseteq \mathsf{msg}(n_1)$.

For each such c, we find the relevant S by examining every $n_0 \prec_{\mathbb{A}_1} n_1$. For each path p such that $p(\mathsf{msg}(n_0)) = c$ and p encounters no key edge, we collect the topmost encryption e such that p traverses e before reaching c. If there is no such e, then c is discarded. We let S be the set of these e.

If $c \dagger^S \mathsf{msg}(n_1)$, then $\mathsf{Cut}(c, S, \mathbb{A}_1)$ is unsolved. Otherwise c is discarded. This process must deliver at least one c, S pair for n_1. The set S consists of encryptions that were ingredients in messages of $\mathbb{A}_1$, so S is finite.[3]

6.3 Destroying a Test, 1

Suppose $\mathsf{Cut}(c, S, \mathbb{A}_1)$ is unsolved, and—for ease of exposition—it has a single minimal node n_1. We consider now the first way to destroy the test, in which we find message homomorphisms α such that $\alpha(c) \odot^{\alpha(S)} \alpha(\mathsf{msg}(n_1))$.

By the definition, there are some paths $p_1, \ldots, p_i$ each p_j of which leads to c in $\mathsf{msg}(n_1)$, but p_j does not encounter a key edge, nor traverse any encryption in S. To destroy the cut, we must find a message homomorphism α such that each path to $\alpha(c)$ in $\alpha(\mathsf{msg}(n_1))$ either encounters a key edge, or traverses a member of $\alpha(S)$.

We do so via unification. That is, for each p_j, there is a set $E_j = \{e_{j1}, \ldots, e_{jk}\}$ which are the encryptions that p_j does traverse. We must (simultaneously) unify some $e_{j\ell}$ with some member of S, for each $j \leq i$. There are at most $(|S| \cdot |E_j|)$ most general ways to succeed for path p_j, and hence at most $(|S| \cdot |E_j|)^i$ ways to succeed for all paths.[4]

The unification ensures that any message homomorphism that destroys the cut is at least as specific as one of the resulting message homomorphisms α. However, $\alpha(\mathsf{msg}(n_1))$ may have occurrences of $\alpha(c)$ at new paths besides the original $p_1, \ldots, p_i$. In this case, we repeat the unification for the new paths to obtain 0 or more refinements of α that successfully destroy the cut.

Since each success furnishes a unifier α, we have a set of homomorphisms $J_\alpha \colon \mathbb{A}_1 \mapsto \alpha(\mathbb{A}_1)$ which destroy the cut in maximally general ways. The latter means that (1) $\mathsf{Cut}(\alpha(c), \alpha(S), J_\alpha(\mathbb{A}_1))$ is not well-defined, and, moreover, (2) if $\mathsf{Cut}(\beta(c), \beta(S), \beta(\mathbb{A}_1))$ is not well-defined, then $\alpha \leq_s \beta$ for one of the α we have computed. Since $\alpha(\mathbb{A}_1)$ may not be a skeleton, but only a preskeleton, we will use the hull operation later.

6.4 Destroying a Test, 2

Turning to the second way to destroy a test, consider the transmission nodes

$$V = \{m_1 \colon m_1 \text{ is a transmission node } \wedge\, n_1 \not\prec_{\mathbb{A}_1} m_1\}.$$

Suppose that $\alpha(c) \dagger^{\alpha(S)} \alpha(\mathsf{msg}(m_1))$. Suppose also that m_1 is the first node along its strand for which this is true; that is, suppose that $m_0 \Rightarrow^+ m_1$ implies

[3] John Ramsdell devised this process for selecting a test.

[4] John Ramsdell devised this repeated unification. It is used again in Section 6.5.

$\alpha(c) \odot^{\alpha(S)} \alpha(\mathsf{msg}(m_0))$. Then, possibly enriching the ordering of $\mathbb{A}_1$ so that $m_1 \prec n_1$, and applying α, we obtain for each such $m_1 \in V$ a preskeleton $\mathbb{B}$ and a map $J_\alpha \colon \mathbb{A}_1 \mapsto \mathbb{B}$.

6.5 Solutions with a Regular Transmission

Suppose that $\mathsf{Cut}(c, S, \mathbb{A}_1)$ is well-defined but unsolved, with a minimal reception node $n_1 \in \mathsf{Cut}(c, S, \mathbb{A}_1)$. A transmission node m_1 can solve some refinement of the cut if it will be minimal in some $\mathsf{Cut}(\alpha(c), \alpha(S), \alpha(\mathbb{A}_1))$. Then m_1 must be of the form $\alpha(\rho \downarrow i)$ for some role $\rho \in \Pi$, and $\alpha(c) \dagger^{\alpha(S)} \mathsf{msg}(m_1)$. Moreover, since we will choose m_1 to be the earliest solution, $\alpha(c) \odot^{\alpha(S)} \mathsf{msg}(m_0)$ for all $m_0 \Rightarrow^+ m_1$.

We may again rely on repeated unification. Consider every transmission node $\rho \downarrow i$, and each path p that does not encounter a key edge in $\mathsf{msg}(\rho \downarrow i)$. If c is unifiable with $p(\mathsf{msg}(\rho \downarrow i))$, then let α be the most general unifier. Consider now each $\rho \downarrow j$ with $j < i$, and consider all p' such that $p'(\alpha(\mathsf{msg}(\rho \downarrow j))) = \alpha(c)$. We want to arrange that each such p' traverses a member of $\alpha(S)$. We may refine α to a (possibly empty) set of most general β such that $\beta(c) \odot^{\beta(S)} \beta(\mathsf{msg}(\rho \downarrow j))$ using the same repeated unification as in Section 6.3.

For this set of successful β, we have the transmission nodes $\beta(\rho \downarrow i) = m_1$ as the most general solutions under Def. 5.3, Clause 1. For each such β, we build a new preskeleton $\mathbb{B}$ by adding the node m_1 to $\beta(\mathbb{A}_1)$; we enrich the ordering so that $m_1 \prec_{\mathbb{B}} n_1$. The message homomorphism β determines a homomorphism $J_\beta \colon \mathbb{A}_1 \mapsto \mathbb{B}$.

6.6 Solutions that Listen for a Key

The solutions under Def. 5.3, Clause 2 are computed directly from the form of c and S. In particular, for each K^{-1} such that some $\{\!|t|\!\}_K \in S$, where $K^{-1} \notin \mathsf{non}_{\mathbb{A}_1}$, we build a solution $\mathbb{B}$ containing a listener strand $\stackrel{K^{-1}}{\rightarrow} \bullet = m_1$. We extend the ordering so that $m_1 \prec_{\mathbb{B}} n_1$, for each n_1 which is $\preceq_{\mathbb{A}_1}$-minimal in $\mathsf{Cut}(c, S, \mathbb{A}_1)$. If $c = \{\!|t|\!\}_K$, another solution is the skeleton with listener strand $\stackrel{K}{\rightarrow} \bullet = m_1$ added, assuming that $K \notin \mathsf{non}_{\mathbb{A}_1}$. For each of these solutions $\mathbb{B}$, we have an embedding $J_{\mathbb{B}}$ of $\mathbb{A}_1$ into $\mathbb{B}$.

6.7 The Cohort of Solutions

Each of the three types of solutions yields 0 or more potential solutions $\mathbb{B}_i$ for $\mathsf{Cut}(c, S, \mathbb{A}_1)$. Each solution consists of a target preskeleton $\mathbb{B}_i$, together with a homomorphism $J_i \colon \mathbb{A}_1 \mapsto \mathbb{B}_i$. We now apply the hull operation to all of the J_i, collecting the successful results as a set $\mathcal{J}$ of homomorphisms.

The set $\mathcal{J}$ is the *cohort* of solutions for $\mathsf{Cut}(c, S, \mathbb{A}_1)$.

If $\mathcal{J}$ is the empty set, then we have discovered that $\mathbb{A}_1$ is dead: It contains a cut that cannot be solved. Otherwise, the result of the step is to augment the

set $\mathcal{F}$ of homomorphisms still to be considered by adding the set

$$\{J \circ H \colon J \in \mathcal{J}\},$$

where H is the homomorphism we chose originally—and removed from $\mathcal{F}$—to start this step.

7 Soundness and Completeness of the Search

In this section, we prove two central facts about the search for shapes. The first (Thm. 7.1) says that a step never discards any homomorphisms that lead to realized skeletons. We also regard this as showing the following. Suppose:

1. $\mathcal{C}_{\mathbb{A}}$ is a characterization for $\mathbb{A}$
2. $\mathcal{J}$ is a cohort for $\mathbb{A}$;
3. For each $J \colon \mathbb{A} \mapsto \mathbb{A}_j$ with $J \in \mathcal{J}$, $\mathcal{C}_j$ is a characterization for $\mathbb{A}_j$.

Then

$$\min(\mathcal{C}_{\mathbb{A}}) \sim \min(\{H \circ J \colon J \in \mathcal{J} \wedge H \in \mathcal{C}_j\}).$$

The second result, Thm. 7.2, says that—modulo a little bookkeeping—one can view any homomorphism to a realized skeleton as consisting of a finite sequence of steps. The bookkeeping may include a nodewise injective homomorphism at the end, and also a surjective map at the beginning that may identify strands.

Theorem 7.1 (Search Soundness) *Suppose that $H \colon \mathbb{A} \mapsto \mathbb{C}$, where $\mathbb{C}$ is realized, and let $\mathcal{J}$ be the cohort of solutions for the unsolved test $\mathsf{Cut}(c, S, \mathbb{A})$. There is a $J \colon \mathbb{A} \mapsto \mathbb{B}$ with $J \in \mathcal{J}$ and a $K \colon \mathbb{B} \mapsto \mathbb{C}$ such that $H = K \circ J$.*

Proof: Let n_1 be an unsolved, $\preceq_{\mathbb{A}}$-minimal member of $\mathsf{Cut}(c, S, \mathbb{A})$. Since $\mathbb{C}$ is realized, $Q = \mathsf{Cut}(\alpha_H(c), \alpha_H(S), \alpha_H(\mathbb{A}))$ is either undefined or solved.

If Q is undefined, then we apply the first type of test destruction, as in Section 6.3. Indeed, since $\phi_H(n_1)$ is not in Q, some member of $\alpha_H(S)$ is traversed on each path p that does not encounter a key edge. Choose p to be a path such that $p(\mathsf{msg}(n_1)) = c$ but p traverses no member of S, and let p_1 be the shortest proper subpath of p such that $p_1(\alpha_H(\mathsf{msg}(n_1))) = \alpha_H(e)$ for $e \in S$. Let γ be the m.g.u. of $p_1(\mathsf{msg}(n_1))$ and e. By Lemma 3.14, Clause 1, there is a universal homomorphism G such that $\gamma \leq_s \alpha_G$. Repeating this process with all such p, we obtain a step satisfying the constraints in Section 6.3.

Assume next that $Q = \mathsf{Cut}(\alpha_H(c), \alpha_H(S), \alpha_H(\mathbb{A}))$ is solved in $\mathbb{C}$, and there is a transmission node of the form

$$m = \phi_H(m_1) \text{ where } m_1 \in \mathsf{nodes}(\mathbb{A}) \text{ and } \alpha_H(c) \dagger^{\alpha_H(S)} \mathsf{msg}(m).$$

In this case, we have a step according to Section 6.4, as we see again using Lemma 3.14, Clause 1, and if $m_1 \not\preceq_{\mathbb{A}} n_1$, we also use Lemma 3.14, Clause 2.

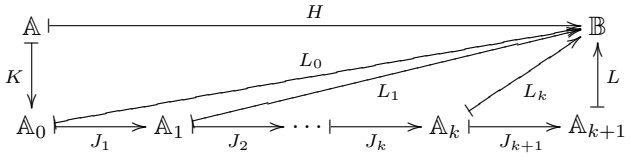


Figure 17: Steps J_i and nodewise-injective L_i

Assume next that there is a transmission node

$$m \in \mathsf{nodes}(\mathbb{C}) \text{ where } \alpha_H(c) \dagger^{\alpha_H(S)} \mathsf{msg}(m), \text{ but } m \notin \phi_H(\mathsf{nodes}(\mathbb{A})).$$

In this case, we have a step according to Section 6.5, where we use Lemma 3.14, Clause 1, to justify any non-trivial message homomorphism.

Otherwise, there is a listener node ℓ for some decryption key inverting an encryption key used in $\alpha_H(S)$, or for the encryption key used in $\alpha_H(c)$. This is the image of a key that may be added in a step according to Section 6.6.

Thus, in all cases, H factors through some step. □

Observe, in this proof, that if a step introduces a node n, and $\phi_K(m)$ lies on the same strand as some $\phi_K(n)$, then already in $\mathbb{B}$, m and n lie on the same strand. No two different strands already present in $\mathbb{B}$ are identified in $\mathbb{C}$. Thus, if H is node-injective, so is the "continuation" homomorphism K. In formulating the next theorem, we rely on Thm. 5.5, Clause 2, to justify the assumption that in $\mathbb{B}$ every well-defined cut is solved.

Theorem 7.2 (Search Completeness) *Suppose that $H\colon \mathbb{A} \mapsto \mathbb{B}$, where every well-defined cut in $\mathbb{B}$ is solved. Then there are:*

1. *A surjective $K\colon \mathbb{A} \mapsto \mathbb{A}_0$;*
2. *A sequence of steps $J_1, \ldots J_{k+1}$ such that each $J_{i+1}\colon \mathbb{A}_i \mapsto \mathbb{A}_{i+1}$ belongs to the cohort for some test $\mathsf{Cut}(c, S, \mathbb{A})_i$; and*
3. *a node-injective $L\colon \mathbb{A}_{k+1} \mapsto \mathbb{B}$*

such that $H = L \circ J_{k+1} \circ \ldots \circ J_1 \circ K$.

Proof: First, we define K to be the universal homomorphism (using Lemma 3.14, Clause 3 repeatedly) that identifies any pair $m, n \in \mathsf{nodes}(\mathbb{A})$ if $\phi_H(m) = \phi_H(n)$. Let $\mathbb{A}_0$ be the target of K. The universality of K implies that H factors (uniquely) through K, i.e. $H = L_0 \circ K$ for some L_0. By the definition of K, L_0 is nodewise injective.

We will define a finite sequence of steps $J_1, \ldots, J_{k+1}$, and refer to the target of J_i as $\mathbb{A}_i$. We do this in such a way that there is always a nodewise injective $L_i\colon \mathbb{A}_i \mapsto \mathbb{B}$, as in Fig. 17.

At each step, given $J_1, \ldots, J_i$, if $\mathbb{A}_i$ is not realized, then we can apply Thm. 7.1 to the map $L_i \colon \mathbb{A}_i \mapsto \mathbb{B}$. We thus split L_i into a step J_{i+1} and a "continuation" homomorphism L_{i+1}. By the remark after the last proof, L_{i+1} is again node-injective.

By node-injectiveness of the L_i, the cardinalities of $\mathsf{nodes}(\mathbb{A}_i)$, of $\preceq_{\mathbb{A}_i}$, of $\mathsf{non}_{\mathbb{A}_i}$, and of $\mathsf{unique}_{\mathbb{A}_i}$ are non-decreasing. By the finiteness of $\mathbb{B}$, only finitely many such $\mathbb{A}_i$ can differ in any of these cardinalities.

Moreover, by Lemma 2.3, Clause 3, there are finitely many non-isomorphic $\beta \leq_s \alpha_{L_0}$. In particular, since $\alpha_{J_i} \circ \ldots \circ \alpha_{J_1} \leq_s \alpha_{L_0}$, there are only finitely many steps possible. Hence, the process terminates with a $\mathbb{A}_{k+1}$ containing no further unsolved cuts, and we let $L = L_{k+1}$. □

Conclusion. We have presented here the theory underlying CPSA, the Cryptographic Protocol Shape Analyzer. We will write more specifically about the design and implementation of CPSA elsewhere.

We believe that the Authentication Test Principle is central to cryptographic protocols. Indeed, already in our first paper about the tests, we pointed out that they provide very strong heuristics to guide cryptographic protocol design [11]. We also illustrated a systematic protocol design process, which led to a protocol achieving goals akin to the Secure Electronic Transaction protocol, organized by reference to the tests [8]. Moreover, we have recently used the tests to justify a criterion for when combining a protocol with new behaviors preserves all security goals met by the original protocol [9]. We hope to develop the theory we have just described to provide a systematic set of protocol transformations that preserve security goals. This appears likely to provide rigorous techniques to replace the heuristics that protocol designers currently use.

Acknowledgments. Thanks especially to Shaddin Dughmi, John D. Ramsdell and F. Javier Thayer, who contributed enormously to developing these ideas and making them work. Shaddin Dughmi (a.k.a. Doghmi) wrote the earliest version of CPSA. John Ramsdell is the author of the current version.

Moses Liskov, Leonard Monk, and Paul Rowe helped bring clarity to the presentation. Jonathan Herzog helped work out the original strand space ideas.

References

[1] Boris Balacheff, Liqun Chen, Siani Pearson, David Plaquin, and Graeme Proudler. *Trusted Computing Platforms: TCPA Technology in Context.* Prentice Hall PTR, Upper Saddle River, NJ, 2003.

[2] Bruno Blanchet. *Vérification automatique de protocoles cryptographiques: modèle formel et modèle calculatoire. Automatic verification of security protocols: formal model and computational model.* Mémoire d'habilitation à diriger des recherches, Université Paris-Dauphine, November 2008.

[3] Cas J.F. Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *ACM Conference on*

Computer and Communications Security (CCS), pages 119–128, New York, NY, USA, 2008. ACM.

[4] C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Ph.D. dissertation, Eindhoven University of Technology, 2006.

[5] Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Completeness of the authentication tests. In J. Biskup and J. Lopez, editors, *European Symposium on Research in Computer Security (ESORICS)*, number 4734 in LNCS, pages 106–121. Springer-Verlag, September 2007.

[6] Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Searching for shapes in cryptographic protocols. In *Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, number 4424 in LNCS, pages 523–538. Springer, March 2007. Extended version at URL:`http://eprint.iacr.org/2006/435`.

[7] Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Skeletons, homomorphisms, and shapes: Characterizing protocol executions. In M. Mislove, editor, *Proceedings, Mathematical Foundations of Program Semantics*, April 2007.

[8] Joshua D. Guttman. Authentication tests and disjoint encryption: a design method for security protocols. *Journal of Computer Security*, 12(3/4):409–433, 2004. Extended version of "Security Protocol Design via Authentication Tests," CSFW 2002.

[9] Joshua D. Guttman. Cryptographic protocol composition via the authentication tests. In Luca de Alfaro, editor, *Foundations of Software Science and Computation Structures (FOSSACS)*, number 5504 in LNCS, pages 303–317. Springer, March 2009.

[10] Joshua D. Guttman. Fair exchange in strand spaces. In M. Boreale and S. Kremer, editors, *SecCo: 7th International Workshop on Security Issues in Concurrency*, EPTCS. Electronic Proceedings in Theoretical Computer Science, Sep 2009.

[11] 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.

[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] 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.

[14] Dag Prawitz. *Natural Deduction: A Proof-Theoretic Study*. Almqvist and Wiksel, Stockholm, 1965.

[15] Dawn Xiaodong Song. Athena: a new efficient automated checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1999.

[16] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.

Contents

List of Figures