The CPSA Specification: A Reduction System for Searching for Shapes in Cryptographic Protocols

John D. Ramsdell Joshua D. Guttman Paul D. Rowe
The MITRE Corporation

November 5, 2010

This is a draft and many of the theorems lack proofs. The draft has an index and a table of contents at its end. Hopefully, the final paper will support an abstract of the following form.

Abstract

We describe a term reduction system that enumerates all essentially different executions possible for a cryptographic protocol. We call them the *shapes* of the protocol. Naturally occurring protocols have only finitely many, indeed very few shapes. Authentication and secrecy properties are easy to determine from them, as are attacks and anomalies. Our Cryptographic Protocols Shapes Analyzer (CPSA) program is a direct implementation of the reduction system described within, and the form of the reduction system is partially determined by the implementation.

The reduction system is a purely syntactic description of an abstract version of the algorithm published previously. Order-sorted term algebras are used for message algebras, and allow us to extend the algorithm to algebras with an associative-commutative operation. During the process of refining the description of the algorithm, we

found a significant simplification of the algorithm and that elaborating some abstractions required great care.

This paper describes the reduction system and shows it possesses desired correctness properties. In particular, its search is *complete* meaning when analyzing a protocol starting with some initial behavior, every shape can be found in a finite number of steps.

Background material should be here.

Initial attempts at implementing the Cryptographic Protocols Shapes Analyzer (CPSA) algorithm in [2] revealed that the strand space framework required extensions and modifications to specify the design and implementation of a CPSA program. The first step was to capture the essence of a strand space algebra in the order-sorted algebra framework. The second step was to choose data structures, and the third step was to specify operations on the data structures that produce the desired results.

When using strand space theory, one normally hypothesizes the existence of a single global strand space. This is a very reasonable assumption for theoretical analysis, but from the point view of an implementer, it turns out that it is better to assume there are many local strand spaces and the design specification task is to describe the relations between these local spaces. Our reformulation of strand space notation provides an implementation oriented way of describing the concept of a local strand space, and a direct link between from algorithm specification to the data structures used in the implementation.

In this specification, each reduction step in the search algorithm relates a term to a set of terms. The correctness properties of the algorithm justify a reduction step by placing restrictions on the relations relative to their respective local strand spaces. The fact that an efficient implementation is available is due in part to the fact the reduction system is confluent, thus greatly reducing the search space.

The sections that follow should described here.

In what follows, a finite sequence is a function from an initial segment of the natural numbers. The length of a sequence f is $|f|$, $f_0 \frown f_1$ is the concatenation of sequences f_0 and f_1, and sequence $f = \langle f(0), \ldots, f(n-1) \rangle$ for $n = |f|$. If S is a set, then S^* is the set of finite sequences of S, and S^+ is the non-empty finite sequences of S.

Section 7 will define the terms "skeleton" and "pruned skeleton". Starting in Section 11, all skeletons are pruned, so "skeleton" is used for pruned

skeletons.

1 Overview

An implementation-oriented view of strand spaces and bundles [6] follows. In this paper, a run of a protocol is viewed as an exchange of messages by a finite set of local sessions of the protocol. Consider the class of Σ, Γ-algebras, algebras over signature Σ that satisfy equations Γ [1, Chapter 3]. A message is an element of $\mathfrak{M}_{\Sigma,\Gamma}(X)$, a free Σ, Γ-algebra generated by set X.

Each local session is called a *strand*, and is an element of the strand set Θ. The behavior of a participant, its *trace*, is a sequence of messaging events. An *event* is either a message transmission or a reception. Outbound message $t \in \mathfrak{M}_{\Sigma,\Gamma}(X)$ is written as $+t$, and inbound message t is written as $-t$. A *strand space* over algebra $\mathfrak{M}_{\Sigma,\Gamma}(X)$ is the strand set Θ together with a trace mapping $tr\colon \Theta \to (\pm\mathfrak{M}_{\Sigma,\Gamma}(X))^+$. In a strand space, the elements of the generator set X denote atomic message elements, such as keys and text data, and not composite messages, such as encryptions and pairs. Later in this paper, strand sets Θ will be initial segments of the natural numbers, so a trace mapping will be a sequence.

Message events occur at nodes in a strand space. For each strand s, there is a node for every message in $tr(s)$. The *nodes* of strand space (Θ, tr) are $\{(s,p) \mid s \in \Theta, 0 \leq p < |\, tr(s)|\}$, and the event at a node is $evt(s,p) = tr(s)(p)$. The relation $\Rightarrow$ defined by $\{(s,p-1) \Rightarrow (s,p) \mid s \in \Theta, 0 < p < |\, tr(s)|\}$ is called the *strand succession relation*.

A *bundle* in strand space (Θ, tr) is a finite directed acyclic graph $\mathcal{B}(\Theta, tr, \rightarrow)$, where the vertices are the nodes of (Θ, tr), and an edge represents communication ($\rightarrow$) or strand succession ($\Rightarrow$). For communication, if $n_0 \rightarrow n_1$, then there is a message t such that $evt(n_0) = +t$ and $evt(n_1) = -t$. For each reception node n_1, there is a unique transmission node n_0 with $n_0 \rightarrow n_1$.

In a run of a protocol, the behavior of each strand is constrained by a role in a protocol. Adversarial strands are constrained by roles as are non-adversarial strands. A role is a sequence of message transmission and reception terms that serves as a template for its strands. Let $\mathfrak{A}_{\Sigma,\Gamma}(Y)$ be a Σ, Γ-term algebra generated by variable set Y. A *role* is a trace in $(\pm\mathfrak{A}_{\Sigma,\Gamma}(Y))^+$. A *protocol* is a set of roles.

A strand space (Θ, tr) *respects protocol* P if there is a role mapping $rl\colon \Theta \to P$ with the following properties. For each strand s, $|\, tr(s)| \leq$

$|\, rl(s)|$, and there is a Σ-homomorphism $\sigma\colon \mathfrak{A}_{\Sigma,\Gamma}(Y) \to \mathfrak{M}_{\Sigma,\Gamma}(X)$ such that $\sigma(rl(s)(p)) = tr(s)(p)$ for all $0 \leq p < |\, tr(s)|$. A bundle $\mathcal{B}(\Theta, tr, \rightarrow)$ is a *run of protocol* P if (Θ, tr) respects protocol P.

In what follows, $\mathfrak{M}_{\Sigma,\Gamma}(X)$ is replaced by term algebra $\mathfrak{A}_{\Sigma,\Gamma}(Z)$, where $|Z| = |X|$, so that the two algebras are isomorphic.

Now, on to the refinement of the definitions in this section. The next section adds structure to message algebras using a sort system. Later on, Definition 5.6 adds information to a role and Definition 6.7 refines the notion of a run of a protocol using the additional information.

2 Messages

The CPSA program implements a message algebra as an order-sorted term algebra [3]. The signature of one of the implemented algebras is given in Figure 1. Sort mesg is the sort of all messages, the elements of the algebras in the previous section.

The use of order-sorted algebras is a partial solution to the following problem. When a strand receives the pair $\mathsf{pair}(t_0, t_1)$, it can extract both t_0, and t_1. When a strand receives the encryption $\mathsf{enc}(t_0, t_1)$, it can extract t_0 if it has the decryption key associated with t_1. And if a strand receives the asymmetric key $\mathsf{invk}(t)$, it can never extract t. How does the CPSA program distinguish these cases? It classifies a term by its sort and the position at which it occurs within another term. The sort system used to classify terms is presented first.

Following [3], the set of messages used by CPSA is specified by an order-sorted signature $(S, \leq, \Sigma)$, where $(S, \leq)$ is a partially ordered set of sort symbols, and Σ is an $S^* \times S$-sorted family $\{\Sigma_{w,s} \mid w \in S^*, s \in S\}$ of operation (or function) symbols. The operations satisfy the following monotonicity condition, $f \in \Sigma_{w,s} \cap \Sigma_{w',s'}$ and $w \leq w'$ imply $s \leq s'$, where $w \leq w'$ iff $|w| = |w'|$ and $w(i) \leq w'(i)$ for $i < |w|$. A variable set X, is an S-sorted family $X = \{X_s \mid s \in S\}$ of disjoint sets of symbols. No symbol is both a variable and an operation. When S and $\leq$ are clear, we write Σ for $(S, \leq, \Sigma)$.

Definition 2.1 (Regular Signature)**.** An order-sorted signature Σ is *regular* iff given $f \in \Sigma_{w',s'}$ and given $w'' \leq w'$ in S^*, there is a least $(w, s) \in S^* \times S$ such that $w'' \leq w$ and $f \in \Sigma_{w,s}$.

Definition 2.2 (Order-Sorted Algebra)**.** Let $(S, \leq, \Sigma)$ be an order-sorted signature. Then an $(S, \leq, \Sigma)$-*algebra* $\mathfrak{A}$ is a family $\{\mathfrak{A}_s \mid s \in S\}$ of sets called the *carriers* of $\mathfrak{A}$, together with a function $\mathfrak{A}_f : \mathfrak{A}_w \to \mathfrak{A}_s$ for each $f \in \Sigma_{w,s}$ where $\mathfrak{A}_w = \mathfrak{A}_{w(0)} \times \cdots \times \mathfrak{A}_{w(n-1)}$, where $n = |w|$. Moreover,

1. $s \leq s'$ in S implies $\mathfrak{A}_s \subseteq \mathfrak{A}_{s'}$ and
2. $f \in \Sigma_{w,s} \cap \Sigma_{w',s'}$ and $w \leq w'$ imply $\mathfrak{A}_f : \mathfrak{A}_w \to \mathfrak{A}_s$ equals $\mathfrak{A}_f : \mathfrak{A}_{w'} \to \mathfrak{A}_{s'}$ on $\mathfrak{A}_w$.

Definition 2.3 (Order-Sorted Terms)**.** The order-sorted Σ-terms $\mathcal{T}_\Sigma(X)$ generated by variable set X is the least family $\{\mathcal{T}_{\Sigma,s}(X) \mid s \in S\}$ such that

1. $\Sigma_{\langle\rangle,s} \subseteq \mathcal{T}_{\Sigma,s}(X)$ and $X_s \subseteq \mathcal{T}_{\Sigma,s}(X)$ for $s \in S$;
2. $\mathcal{T}_{\Sigma,s'}(X) \subseteq \mathcal{T}_{\Sigma,s}(X)$ if $s' \leq s$;
3. if $f \in \Sigma_{w,s}$ and $t_i \in \mathcal{T}_{\Sigma,w(i)}(X)$ for $i < |w|$, then $f(t_0, \ldots, t_{|w|-1}) \in \mathcal{T}_{\Sigma,s}(X)$.

In [3], it is proved that $\mathcal{T}_\Sigma(X)$ is an order-sorted algebra and when Σ is regular, each term has a least sort.

An *equation* is a triple (X, t, t'), written $t \approx t'$, where X is a variable set and t, t' are in $\mathcal{T}_\Sigma(X)$ and have the same least sort. See [3] on how to relax the sort restriction. The review of order-sorted algebras ends here.

Definition 2.4 (Encryption Signature)**.** A regular order-sorted signature is an *encryption signature* if it contains a distinguished sort symbol $\top$, the sort of all messages, and a binary encryption operation enc defined so that the least sort of every term of the form $\mathsf{enc}(t_0, t_1)$ is $\top$.

Definition 2.5 (Strand Space Signature)**.** A *strand space signature* (Σ, B) is an encryption signature Σ, and a non-empty subset B of sort symbols in Σ such that $s \in B$ implies $s < \top$. Each member of B is called a *base sort*. Every operation in Σ with a base sorted result must have base sorted arguments.

Definition 2.6 (Strand Space Equations)**.** Equations in Γ for strand space signature (Σ, B) are *strand space equations* if the least sort of the terms that define each equation in Γ is a base sort in B.

Base sort symbols: name, text, data, skey, akey
Non-base sort symbol: mesg (implementation of $\top$)

Subsorts: name, text, data, akey, skey $<$ mesg

$\mathsf{enc}\colon \mathrm{mesg} \times \mathrm{mesg} \to \mathrm{mesg}$	Encryption
$\mathsf{pair}\colon \mathrm{mesg} \times \mathrm{mesg} \to \mathrm{mesg}$	Pairing
$C_i\colon \mathrm{mesg}$	Tag constants ($i \in \mathbb{N}$)
$\mathsf{pubk}\colon \mathrm{name} \to \mathrm{akey}$	Public key of name
$\mathsf{invk}\colon \mathrm{akey} \to \mathrm{akey}$	Inverse of asymmetric key
$\mathsf{ltk}\colon \mathrm{name} \times \mathrm{name} \to \mathrm{skey}$	Long term shared key

Equation: $\mathsf{invk}(\mathsf{invk}(x)) \approx x$ for $x\colon$ akey

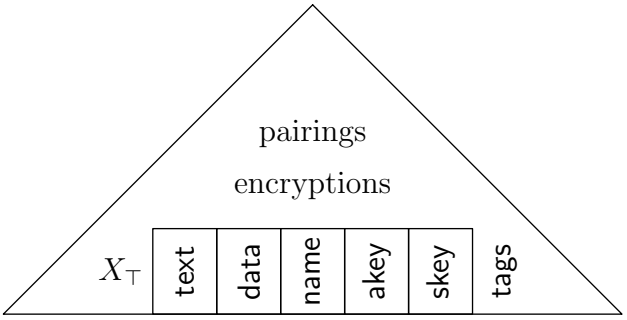
Figure 1: Basic Crypto Signature and Equation

The unification type of Γ w.r.t. signature Σ is *unitary* if every problem has a minimal complete set of unifiers that has at most one element [5, Chapter 8, Section 3.1].

Definition 2.7 (Strand Space Algebra)**.** Given strand space signature (Σ, B) and strand space equations Γ, a *strand space algebra* $\mathfrak{A}_{\Sigma,B,\Gamma}(X)$ is the quotient of $\mathcal{T}_\Sigma(X)$ by the equivalence relation on $\mathcal{T}_\Sigma(X)$ derived from the equations in Γ, where the unification type of Γ w.r.t. signature Σ is unitary.

In the remainder, the strand space signature and equations are fixed, and its algebras are written as $\mathfrak{A}(X)$ and as $\mathfrak{A}$ when the variable set X is available from the context, and the term algebras as $\mathcal{T}(X)$ and $\mathcal{T}$ respectively. Relation $\equiv$ is the equivalence relation on $\mathcal{T}_\top$ derived from the equations in Γ, and the equivalence class $[t]$ is $\{t' \mid t \equiv t'\}$. We assume each equivalence class has a canonical representative. At times we conflate a term with the equivalence class of which it is a member to simplify the presentation. Given a sort s, $t\colon s$ asserts that $[t] \in \mathfrak{A}_s$, and $t\colon S$ asserts that for some $s \in S$, $t\colon s$. Term t is a *message* if $[t] \in \mathfrak{A}_\top$. Message t is an *atom* iff $t\colon B$. The message $\mathsf{enc}(t_0, t_1)$ is written $\{|t_0|\}_{t_1}$.

In the Basic Crypto algebra, sort mesg implements sort $\top$, and all of its other sorts are base sorts. A term of the form $\mathsf{invk}(t)$ is an atom. Figure 2 shows the classification of terms in the Basic Crypto algebra. In text, we use comma as a right associative binary operation for pairing and K_a for $\mathsf{pubk}(a)$, as in the caption of Figure 4.



Carrier set of sort mesg
$X_\top$: variables of sort mesg

Figure 2: Basic Crypto Carrier Sets

New base sorts:	base, expn $<$ mesg	
New operations:	gen: base	Generator
	exp: base $\times$ expn $\rightarrow$ base	Exponentiation
	mul: expn $\times$ expn $\rightarrow$ expn	Multiplication
	one: expn	Unit
	rec: expn $\rightarrow$ expn	Reciprocal
New equations:	$\mathsf{exp}(\mathsf{exp}(g,x),y) \approx \mathsf{exp}(g,\mathsf{mul}(x,y))$	
	$\mathsf{exp}(g,\mathsf{one}) \approx g$	
	$\mathsf{mul}(x,y) \approx \mathsf{mul}(y,x)$	Commutativity
	$\mathsf{mul}(x,\mathsf{mul}(y,z)) \approx \mathsf{mul}(\mathsf{mul}(x,y),z)$	Associativity
	$\mathsf{mul}(x,\mathsf{one}) \approx x$	Identity
	$\mathsf{mul}(x,\mathsf{rec}(x)) \approx \mathsf{one}$	Cancellation

Figure 3: Additions for Diffie-Hellman Key Exchange

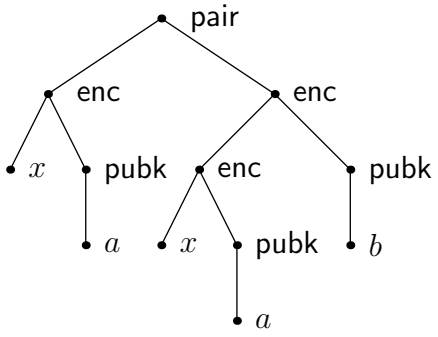


Figure 4: Tree for $\{\!|x|\!\}_{K_a}, \{\!|\{\!|x|\!\}_{K_a}|\!\}_{K_b}$

Theorem 2.1 (Encryption Freely Generated)**.** Suppose $t, t_0, t_1 \in \mathcal{T}_\top$ and $\{\!|t_0|\!\}_{t_1} \equiv t$. Then there exists t'_0, t'_1 such that $t = \{\!|t'_0|\!\}_{t'_1}$, $t_0 \equiv t'_0$, and $t_1 \equiv t'_1$.

Proof. Consider $\mathcal{T}^0_\top(X) \subseteq \mathcal{T}_\top(X)$, where $\mathcal{T}^0_\top(X) = \Sigma_{\langle\rangle,\top} \cup X_\top \cup \bigcup_{s\in B} \mathcal{T}_s(X)$. Observe that the encryption operation does not occur in $\mathcal{T}^0_\top(X)$ and that $\mathcal{T}^0_\top(X)$ freely generates $\mathcal{T}_\top(X)$. Thus $\mathfrak{A}^0_\top(X) = \{[t] \mid t \in \mathcal{T}^0_\top(X)\}$ freely generates $\mathfrak{A}_\top(X)$. No equation applies to messages in $\mathfrak{A}_\top(X) \setminus \mathfrak{A}^0_\top(X)$ and so induction on term formation gives the result. □

Figure 2 shows the carrier sets for the Basic Crypto Algebra.

Definition 2.8 (Position)**.** A *position* p is a finite sequence of natural numbers. The term in t that *occurs at* p, written $t \,@\, p$, is:

$$t \,@\, \langle\rangle = t;$$
$$f(t_0, \ldots, t_{|w|-1}) \,@\, \langle i\rangle \frown p = t_i \,@\, p \text{ if } f \in \Sigma_{w,s} \text{ and } i < |w|.$$

A term t *occurs in* $t' \in \mathcal{T}_\top(X)$ if $t = t' \,@\, p$ for some p. A message t *occurs in* $t' \in \mathfrak{A}_\top(X)$ if the canonical representative of t occurs in the canonical representative of t'.

Some positions within a message are distinguished. A position p *traverses a key edge* in t if $t \,@\, p_0 = \{\!|t_0|\!\}_{t_1}$ and $p = p_0 \frown \langle 1\rangle \frown p_1$. A position p *traverses an atom edge* in t if $t \,@\, p_0 = f(t_0, \ldots, t_{|w|-1})$ is an atom and $p = p_0 \frown \langle i\rangle \frown p_1$, where $f \in \Sigma_{w,s}$ and $i < |w|$. A position is a *carried position* in t if it traverses no key or atom edge in t. In Figure 4, variable x occurs at two carried positions, but a occurs at no carried positions.

The concept of an accessible term is used in Definition 7.9.

Definition 2.9 (Accessible)**.** Message t_0 is *accessible in* t_1, written $t_0 \ll t_1$, if t_0 occurs at a position in t_1 that traverses no atom edge.

A carried term is one that can be extracted from a message reception given the right set of decryption keys.

Definition 2.10 (Carries)**.** Message t_0 *carries* t_1, written $t_1 \sqsubseteq t_0$, if t_1 occurs at a carried position in t_0.

Carried positions respect equality modulo the equations.

Definition 2.11 (Carried Positions)**.** Given a term t, the set of positions at which t' carries t is $\mathit{carpos}(t, t')$, where

$$\mathit{carpos}(t, t') = \begin{cases} \{\langle\rangle\} & \text{if } t' \equiv t, \text{ else} \\ \{\langle 0\rangle \frown p \mid p \in \mathit{carpos}(t, t_0)\} & \\ \qquad & \text{if } t' = \{\!|t_0|\!\}_{t_1}, \text{ else} \\ \{\langle i\rangle \frown p \mid p \in \mathit{carpos}(t, t_i), i < n\} & \\ \qquad & \text{if } t' = f(t_0, \ldots, t_{n-1}) \text{ and} \\ \qquad & t' \text{ is not an atom, else} \\ \{\} & \text{otherwise.} \end{cases}$$

Note that $\mathit{carpos}(x, \mathsf{invk}(\mathsf{invk}(x))) = \{\langle\rangle\}$, not $\{\langle\rangle, \langle 0, 0\rangle\}$, and t' carries t iff $\mathit{carpos}(t, t')$ is non-empty.

Theorem 2.2. For all $t_0, t_1, t_0', t_1' \in \mathcal{T}_\top(X)$, $t_0 \equiv t_1$ and $t_0' \equiv t_1'$ implies $\mathit{carpos}(t_0, t_0') = \mathit{carpos}(t_1, t_1')$.

Proof. When t_0' and t_1' are atoms, the result is obvious. Otherwise, Theorem 2.1 applies. □

The key insight discovered while implementing strand space algebras is that ordinary order-sorted algebras with restricted signatures, coupled with the *carpos* function, captures the essence of strand space algebras required to implement the CPSA algorithm. Section 10 will explain why the details of the *carpos* function are important, and why an implementation of the carries relation does not suffice.

Definition 2.12 (Strand Space with Asymmetric Encryption)**.** A strand space algebra $\mathfrak{A}_{\Sigma,B,\Gamma}(X)$ with asymmetric encryption has a signature with a distinguished base sort symbol akey, an operation $\mathsf{invk}\colon \mathsf{akey} \to \mathsf{akey}$, and Γ contains the equation $\mathsf{invk}(\mathsf{invk}(x)) \approx x$ for $x\colon$ akey.

Definition 2.13 (Inverse Key)**.** For a strand space with asymmetric encryption, the decryption key of encryption $\{|t_0|\}_{t_1}$ is $inv(t_1)$, where

$$inv(t) = \begin{cases} \mathsf{invk}(t) & \text{if } t\colon \text{akey;} \\ \text{undefined} & \text{if } t \text{ is a variable of sort } \top; \\ t & \text{otherwise.} \end{cases}$$

Both unification and matching are used to solve equations. First, a few more definitions from [3] and [5]. Given two S-sorted variable sets X and Y, an *order-sorted substitution* is an S-sorted map $\sigma\colon X \to \mathfrak{A}(Y)$ such that $\sigma(x) \neq x$ for only finite many elements of X. For a substitution σ, the *domain* is the set of variables $Dom(\sigma) = \{x \mid \sigma(x) \neq x\}$ and the *range* is the set of terms $Ran(\sigma) = \{\sigma(x) \mid x \in Dom(\sigma)\}$. Substitution σ_0 is *more general than* σ_1, written $\sigma_0 \trianglelefteq \sigma_1$, if there exists a substitution σ_2 such that $\sigma_1(x) \equiv \sigma_2(\sigma_0(x))$ for $x \in X$, the variable set that generates the term algebra. It is possible to have $\sigma_0 \trianglelefteq \sigma_1$ and $\sigma_1 \trianglelefteq \sigma_0$. In this case, $\sigma_1 = \sigma_2 \circ \sigma_0$ and $\sigma_0 = \sigma_2' \circ \sigma_0$ where σ_2 and σ_2' are renamings. That is, $Dom(\sigma_2) = Ran(\sigma_2)$ and likewise for σ_2'. In this case we say that σ_0 and σ_1 are renamings of each other. Given an S-sorted map $\varphi\colon X \to \mathfrak{A}(Y)$, the unique order-sorted Σ-homomorphism $\varphi^*\colon \mathfrak{A}(X) \to \mathfrak{A}(Y)$ induced by φ is also denoted φ.

The algorithms used to solve unification and matching problems typically solve many equations at once. To facilitate this requirement, algebras provide functions with the following signatures.

$$\begin{aligned} &\mathit{unify}\colon \mathcal{T}_\top(X) \times \mathcal{T}_\top(X) \times (X \to \mathcal{T}_\top(X)) \to (X \to \mathcal{T}_\top(X))^* \\ &\mathit{match}\colon \mathcal{T}_\top(X) \times \mathcal{T}_\top(Y) \times (X \to \mathcal{T}_\top(Y)) \to (X \to \mathcal{T}_\top(Y))^* \end{aligned}$$

Suppose σ_0 has the property that $\sigma_0(t_i) \equiv \sigma_0(t_i')$ for $i < n$. Then $\sigma_1(t_i) \equiv \sigma_1(t_i')$ for $i \leq n$ if $\sigma_1 \in \mathit{unify}(t_n, t_n', \sigma_0)$. The *unify* function has this property:

$$\mathit{unify}(t_0, t_1, \sigma_0) = \{\sigma_1 \circ \sigma_0 \mid \sigma_1 \in \mathit{unify}(\sigma_0(t_0), \sigma_0(t_1), \sigma_{\mathrm{id}})\},$$

where σ_{id} is the identity substitution. Furthermore, it must produce a minimal complete set of unifiers.

Lemma 2.3 (Order of Unification)**.** Suppose that the underlying message algebra has single most general unifiers for any two unifiable terms. Suppose also that $\mathit{unify}(t_0, t_1, \sigma_{\mathrm{id}}) = \{\sigma_0\}$ and that $\mathit{unify}(s_0, s_1, \sigma_{\mathrm{id}}) = \{\sigma_1\}$. Then $\mathit{unify}(s_0, s_1, \sigma_0) \cong \mathit{unify}(t_0, t_1, \sigma_1)$. That is, when one side exists so does the other, and the two substitutions will be renamings of each other.

A proof of this result can be found for example in [1]. By a simple induction, given any system of equations, $E = \{t_1 \stackrel{?}{=} t'_1, \ldots, t_n \stackrel{?}{=} t'_n\}$, we may solve them in any order and we will end up with the same substitution (up to renaming).

This justifies more general definitions that will be used later. Let A_0 be a set of terms which you would like to jointly unify. That is, A_0 represents the equations $\{t_i \stackrel{?}{=} t_j \mid t_i, t_j \in A_0\}$. Then we may define $\mathit{Unif}(A_0, \sigma) = \mathit{unify}(t_{n-1}, t_n, \mathit{unify}(\ldots, \mathit{unify}(t_0, t_1, \sigma)\ldots))$. By Lemma 2.3 this definition does not depend on the order of the terms $t_0, \ldots, t_n$. The result will be the same up to renaming for any ordering of the terms. It then follows by further induction that $\mathit{Unif}(A_1, \mathit{Unif}(A_0, \sigma)) \cong \mathit{Unif}(A_0, \mathit{Unif}(A_1, \sigma))$. This justifies the more general definition

$$U(A_0, \ldots, A_n; \sigma) = \mathit{Unif}(A_n, \mathit{Unif}(\ldots, \mathit{Unif}(A_0, \sigma)\ldots)).$$

Again we could have defined it in terms of any order of the sets $A_0, \ldots, A_n$. The result might be a different substitution but it would be a renaming of the one defined.

Discussion

The message algebra that appears in strand space papers [6, 4] was not implemented because there is no syntactic method that can be used to determine if an encryption denotes symmetric or asymmetric encryption. The signature in Figure 1 resolves this problem.

The OSA conjecture of this paper is that the results in every strand space paper would remain unchanged if it used algebras from Definition 2.7 as long as unification in the algebra produces at most one most general unifier. Notice there is no prohibition on constants being atoms in the definition of a strand space algebra in this paper, in contrast with the algebras defined in other papers.

Readers of this paper are encouraged to consider other algebras, such as those that contain an Abelian group and modular exponentiation. Such algebras allow the analysis of Diffie-Hellman Key Exchange, for example. See Figure 3.

Some strand space papers defined the word occurs to mean carried by. This document uses it to assert a message is within another message.

Additional sort symbols: atom, evt, role, maplet,
instance, node, ordering, and preskel

Subsorts: for each $s \in B, s \leq \text{atom} < \text{mesg}$

$+ \colon \text{mesg} \to \text{evt} \quad - \colon \text{mesg} \to \text{evt} \quad \mathsf{r} \colon \text{evt list} \times \text{atom set} \times \text{atom set} \to \text{role}$
$\mathsf{m} \colon \text{mesg} \times \text{mesg} \to \text{maplet} \quad \mathsf{i} \colon \text{role} \times \text{nat} \times \text{maplet set} \to \text{instance}$
$\mathsf{n} \colon \text{nat} \times \text{nat} \to \text{node} \quad \mathsf{o} \colon \text{node} \times \text{node} \to \text{ordering}$
$\mathsf{k} \colon \text{role set} \times \text{instance list} \times \text{ordering set} \times \text{atom set} \times \text{atom set} \to \text{preskel}$

mesg	the sort of all messages (implementation of $\top$)
atom	the sort of all base sorted messages
evt	a transmission or reception event
trace	a sequence of events used in a role
role	a trace, a non-originating set, and a uniquely-originating set
protocol	a set of roles
nat	a natural number
maplet	a map from a role variable to a preskeleton term
instance	a strand's trace and inheritance as instantiated from a role
node	a pair of numbers, a strand identifier and a strand position
ordering	a causal ordering between a pair of nodes
preskel	a preskeleton

Table 1: CPSA Signature

3 Data Structures

The data structures used in the implementation are modeled as elements in an order-sorted term algebra. The signature for the algebra is the extension of a strand space signature shown in Table 1. In this specification, it is assumed the strand space signature uses the sort symbol mesg to implement sort $\top$.

Every element of the sort atom is intended to be an element of some base sort. Furthermore, elements of the other sorts added by a CPSA signature correctly model an implementation when there are no variables of those sorts. As a result, the algebras that model an implementation are CPSA algebras.

Definition 3.1 (CPSA Algebra)**.** Consider the class of Σ, Γ-algebras, algebras over CPSA signature Σ that satisfy equations Γ. Algebra $\mathfrak{C}(X)$ is a CPSA *algebra* if it is a free Σ, Γ-algebra generated by variable set X, and the variable

set X has the following property. For sort s, X_s is empty when s is atom, evt, role, maplet, instance, node, ordering, and preskel.

Section 5 on protocols defines events (evt), traces (env list), and roles (role). Section 6 on executions defines instances (instance) and role substitutions (maplet set). Section 7 on skeletons defines, nodes (node), node orderings (ordering set), and preskeletons (preskel).

4 Algorithms as Term Reduction Systems

Algorithms in this paper are specified as abstract reduction systems [1, Chapter 2]. A reduction system is a pair $(A, \rightarrow)$, where reduction $\rightarrow$ is a binary relation $\rightarrow \subseteq A \times A$. Element $x \in A$ is a *normal form* if there is no y such that $x \rightarrow y$. The transitive closure of $\rightarrow$ is $\rightarrow^+$. The reflexive transitive closure of $\rightarrow$ is $\rightarrow^*$. A reduction is confluent if $x \rightarrow^* y_0$ and $x \rightarrow^* y_1$ implies there is a z such that $y_0 \rightarrow^* z$ and $y_1 \rightarrow^* z$. A reduction is terminating if there are no infinite descending chains. A reduction is convergent if it is confluent and terminating.

Let $\mathfrak{K}$ be $\mathfrak{A}_{\mathrm{preskel}}(X)$. Algorithms are specified as reduction systems of the form $(\mathfrak{K}, \rightarrow)$, which are then used to specify a related setwise reduction system of the form $(2^{\mathfrak{K}}, \twoheadrightarrow)$. Setwise reduction systems are the ones with the interesting normal forms and confluence properties. In a setwise reduction system, reduction rewrites one element of a set to a set of elements.

Definition 4.1 (Setwise Reduction System)**.** The *setwise reduction system* of binary relation $\rightsquigarrow \subseteq \mathfrak{K} \times 2^{\mathfrak{K}}$ is a reduction system $(2^{\mathfrak{K}}, \twoheadrightarrow)$, where for each $K_0 \in 2^{\mathfrak{K}}$, $K_0 \twoheadrightarrow K_1$ if for some $k_0 \in K_0$, $k_0 \rightsquigarrow K_2$, $K_1 = K_2 \cup (K_0 \setminus \{k_0\})$, and $K_1 \neq K_0$.

The CPSA algorithm will be specified as a setwise term reduction system, where the initial problem is given a singleton in $2^{\mathfrak{K}}$, and the answers computed by an implementation of the algorithm are a normal form of the setwise reduction relation $\twoheadrightarrow_k$ defined in Section 15.

In what follows the relation $k \rightsquigarrow K$ is defined in terms of $\rightarrow \subseteq \mathfrak{K} \times \mathfrak{K}$ by specifying $\{k\} \twoheadrightarrow K$ using $\rightarrow$, so the $\rightsquigarrow$ relation is not explicitly defined.

We regard sets of preskeletons as factored by isomorphism, where each set has at most one representative of the equivalence class of isomorphic preskeletons. The definition of isomorphic preskeletons is given in Definition 7.4.

5 Protocols

A protocol defines the patterns of allowed behavior for each participant in an execution of the protocol. Protocol participants send and receive messages.

Definition 5.1 (Event)**.** An *event* is either a message transmission or a reception. Formally, an event is a pair (d, t) with $t \in \mathfrak{A}_\top$ and d one of the symbols $+$ or $-$. A positive message $+t$ is *outbound*, and a negative message $-t$ is *inbound.*

In a CPSA algebra, an event is a term of sort evt.

Definition 5.2 (Trace)**.** A *trace* is a non-empty sequence of events, an element of $(\pm\mathfrak{A}_\top)^+$

In a CPSA algebra, a trace is a term of sort evt list.

Definition 5.3 (Originates)**.** A message *originates* in a trace if it is carried by some event and the first event in which it is carried is outbound.

Definition 5.4 (Gained)**.** A message is *gained* in a trace if it is carried by some event and the first event in which it is carried is outbound.

Definition 5.5 (Acquired)**.** A message is *acquired* by a trace if it first occurs in an inbound message and is also carried by that message.

The next definition describes syntactic constraints on "uniquely-originating" and "non-originating" atoms. The meaning of these adjectives is not revealed until Definition 6.7.

Definition 5.6 (Role)**.** A *role* r consists of a trace $\mathit{rtrace}(r)$, a set of non-originating atoms $\mathit{rnon}(r)$, and a set of uniquely originating atoms $\mathit{runique}(r)$. The variable set $\mathit{rvar}(r)$ of the role is the set of variables that occur in $\mathit{rtrace}(r)$. The following properties hold.

1. Each uniquely originating atom originates in the trace.
2. Each non-originating atom is not carried by any event in the trace, and each variable that occurs in the atom occurs in the trace.
3. Every non-base sorted variable is acquired by the trace.

In a CPSA algebra, a role is a term of the form $\mathsf{r}(C, N, U)$, where

$$\begin{aligned} \mathit{rtrace}(\mathsf{r}(C, N, U)) &= C, \\ \mathit{rnon}(\mathsf{r}(C, N, U)) &= N, \\ \mathit{runique}(\mathsf{r}(C, N, U)) &= U. \end{aligned}$$

Definition 5.7 (Protocol)**.** A *protocol* is a set of roles with pairwise disjoint variable sets.

Discussion

The refinement of strand space protocols is mostly just a change in representation for the same information. The change is motivated by the way protocols are used. In this refinement, a protocol is defined without reference to the concept of a strand, a node, or a strand space.

In [2], a protocol is a triple $(\Pi, \mathit{strand\_non}, \mathit{strand\_unique})$, where Π is a finite set of strands, *strand_non* and *strand_unique* map a strand to a set of atoms, and the mappings have properties analogous to the ones in the Definition 5.6, the definition of a role. The set of roles is $\{r \mid s \in \Pi, \mathit{rtrace}(r) = \mathit{tr}(s), \mathit{runique}(r) = \mathit{strand\_unique}(s), \mathit{rnon}(r) = \mathit{strand\_non}(s)\}$.

In [2], non-originating messages were assumed to be keys of the form K or K^{-1}, thereby excluding long term shared symmetric keys of the form $\mathsf{ltk}(A, B)$. The definition of role allows symmetric keys to be assumed to be non-originating.

6 Executions

Executions of a protocol are formalized by a bundle, which is described in this section. A key difference in this approach to formalizing strand spaces is that the parameters used to instantiate a strand's trace from a role are preserved, so as to support role origination assumptions.

Definition 6.1 (Instance)**.** An *instance* i consists of a role $\mathit{role}(i)$, a positive number $\mathit{height}(i)$, and an order-sorted substitution $\mathit{subst}(i)$. Let r, h, and σ be the role, height, and substitution for a given instance i. Let $C|_h$ be the prefix of sequence C of length h. The following properties hold.

1. The height of an instance cannot exceed the length of it's role's trace.

2. $Dom(\sigma)$ is the set of variables that occur in $rtrace(r)|_h$.

3. No variable in an instance's role may occur in $Ran(\sigma)$.

In a CPSA algebra, an instance is a term of the form $\mathsf{i}(r, h, M)$ where

$$\begin{aligned} role(\mathsf{i}(r, h, M)) &= r, \\ height(\mathsf{i}(r, h, M)) &= h, \\ subst(\mathsf{i}(r, h, M)) &= \sigma \end{aligned}$$

and $\sigma(x) = y$ for each $\mathsf{m}(x, y) \in M$.

Definition 6.2 (Trace of Instance)**.** The trace of an instance i, $trace(i)$, is a sequence of length $height(i)$ such that for $\sigma = subst(i)$ and all $j < height(i)$, $trace(i)(j) = \sigma(rtrace(role(i))(j))$.

For every instance i, the definitions imply that the set of variables that occur in $trace(i)$ is a subset of the set of variables that occur in $Ran(subst(i))$. The Diffie-Hellman algebra provides an example in which the subset relation is proper. Consider a role with a trace of $\langle \mathsf{exp}(x, \mathsf{mul}(y, z)) \rangle$. The definitions allow an instance of the role with height one and substitution $\{x \mapsto a, y \mapsto \mathsf{mul}(b, d), z \mapsto \mathsf{mul}(c, \mathsf{rec}(d))\}$, giving the instance a trace of $\langle \mathsf{exp}(a, \mathsf{mul}(b, c)) \rangle$.

Definition 6.3 (Instance Origination Assumptions)**.** Let $\sigma = subst(i)$ for an instance i, $r = role(i)$, and $h = height(i)$. Instance i inherits the non-origination assumption $\sigma(t)$ if $t \in rnon(r)$, and the variables in t are in $Dom(\sigma)$. Instance i inherits the unique origination assumption $\sigma(t)$ if $t \in runique(r)$, and t originates in $rtrace(r)|_h$.

Definition 6.4 (Strand Space [6, Definition 2.2])**.** A *strand space* over algebra $\mathfrak{A}_\top$ is a set Θ together with a trace mapping $tr \colon \Theta \to (\pm\mathfrak{A}_\top)^+$.

In this document, a strand set Θ is an initial segment of the natural numbers, so a trace mapping is a sequence. Thus a strand space is the trace sequence $tr : ((\pm\mathfrak{A}_\top)^+)^+$, and its strand set is the domain of tr.

A finite sequence of instances I is a *protocol respecting strand space*. The strand space of I is $tr = trace \circ I$. By construction, the trace of each strand is an instance of a role.

The variable set $var(I)$ is the set of variables that occur in the range of the substitution of its instances. In what follows, definitions are simplified by assuming the variable set associated with an instance sequence is disjoint from the set of variables that occur in its roles.

Definition 6.5 (Strand Space Nodes)**.** The set of *nodes* of I is $nodes(I) = \{(s,p) \mid s < |I|, p < height(I(s))\}$. The event at a given node is $evt(I,(s,p)) = trace(I(s))(p)$. A node is *transmitting* if its event is outbound, otherwise it is *receiving*. The message at a node is written $msg(I,n)$.

Definition 6.6 (Strand Succession)**.** The *strand succession relation* $\Rightarrow$ is $\{(s,p) \Rightarrow (s,p+1) \mid s < |I|, p < height(I(s)) - 1\}$.

Definition 6.7 (Bundle)**.** A *protocol respecting bundle* $\mathcal{B}(I,\rightarrow)$ is a directed acyclic graph, where the vertices are the nodes of I, and an edge represents communication ($\rightarrow$) or strand succession ($\Rightarrow$). For communication, $n_0 \rightarrow n_1$ only if there is a message t such that $evt(I,n_0) = +t$ and $evt(I,n_1) = -t$. For each reception node n_1, there is a unique transmission node n_0 with $n_0 \rightarrow n_1$. Finally, each inherited uniquely originating atom originates on its trace and on no other, and no event in a trace carries an inherited non-originating atom.

Notice that no non-base sorted variables may occur in the trace of an instance in a bundle.

Definition 6.8 (Causal Order)**.** Each acyclic graph has a transitive asymmetric relation $\prec$ on its vertices. The relation specifies the causal ordering of nodes in a bundle. Relation R on set S is *asymmetric* iff $x\,R\,y$ implies not $y\,R\,x$ for all distinct $x, y \in S$.

Definition 6.9 (Unique Origination)**.** An atom uniquely originates in a execution if it originates on exactly one trace.

Definition 6.10 (Non-Originating)**.** An atom is non-originating in an execution if the atom originates in no trace, but each of its variables occurs in some trace.

Discussion

Strand space papers use the word positive to describe an outbound message or a transmitting node, and the word negative to describe an inbound message or a receiving node. The adjectives used in this paper were selected because they are mnemonic.

$base(t) = \langle +t \rangle$, where t is an atom
$tag(t) = \langle +t \rangle$, where t is a tag
$cat(t_0, t_1) = \langle -t_0, -t_1, +(t_0, t_1) \rangle$
$sep(t_0, t_1) = \langle -(t_0, t_1), +t_0, +t_1 \rangle$
$enc(t_0, t_1) = \langle -t_0, -t_1, +\{|t_0|\}_{t_1} \rangle$
$dec(t_0, t_1) = \langle -\{|t_0|\}_{t_1}, -t_2, +t_0 \rangle$, where $t_2 = inv(t_1)$

Figure 5: Penetrator Traces

7 Skeletons

Strands in executions represent both adversarial and non-adversarial behaviors. A strand that represents adversarial behavior is called a *penetrator* strand. The roles available to a penetrator are determined by the message signature. For the Basic Crypto Signature in Figure 1, the traces of the roles are in Figure 5. Penetrator roles make no origination assumptions.

A non-adversarial strand is called *regular*. A typical protocol contains a small finite set of roles used by regular strand. In addition, a regular strand may be an instance of a listener role. For a given message t, a listener's trace is $\langle -t, +t \rangle$. A listener strand is used to assert that a message is not a secret and is available from the penetrator.

The CPSA program uses a skeleton to represent the regular behavior that might make up part of an execution.

Definition 7.1 (Preskeleton)**.** A *preskeleton* k consists of instance sequence $insts(k)$, a transitive asymmetric node ordering $\prec_k$, a set of uniquely originating atoms $unique(k)$, and a set of non-originating atoms $non(k)$. The following properties hold.

1. The relation $\prec_k$ includes strand succession ($\Rightarrow$).

2. Each atom in $unique(k)$ is carried in the trace of some instance in $insts(k)$.

3. Every inherited unique origination assumption is in $unique(k)$, each inherited atom originates on the inheriting strand, and the origination position of the inherited atom is the same as the origination position of the inherited atom in the strand's trace.

4. Each atom in $non(k)$ is not carried by an event in the trace of some instance in $insts(k)$, and each of variable that occurs in the atom occurs in some trace.

5. Every inherited non-origination assumption is in $non(k)$.

The set $\mathfrak{K}(X)$ is the set of preskeletons k such that $var(insts(k)) \subseteq X$, written $\mathfrak{K}$ when the variable set X is available from the context. To simplify notation, let $ht(k, s) = height(inst(k)(s))$.

In a CPSA algebra, a preskeleton is a term of the form $\mathsf{k}(P, I, O, N, U)$ where

$$\begin{aligned} insts(\mathsf{k}(P, I, O, N, U)) &= I, \\ non(\mathsf{k}(P, I, O, N, U)) &= N, \\ unique(\mathsf{k}(P, I, O, N, U)) &= U. \end{aligned}$$

The implementation of a preskeleton keeps track of its protocol as a set of roles, P, but we ignore the protocol here as well as the fact in a well formed preskeleton, the role of every instance is an element of P. The implementation of a preskeleton node ordering is not so obvious. For preskeleton k, only a subset of $\prec_k$ is explicit: $\mathsf{o}(n_0, n_1) \in O$ if $n_0 \prec_k n_1$, n_0 and n_1 are on different strands, n_0 is transmitting, and n_1 is receiving.

To ease the task of isomorphism testing (Section 8) and generalization by weakening (Section 13), the implementations normalizes a preskeleton by performing the transitive reduction on O. The transitive reduction of a relation is the minimal relation such that both have the same transitive closure.

Definition 7.2 (Hulled Preskeleton)**.** A preskeleton k is a *hulled preskeleton* if each atom in $unique(k)$ originates in at most one trace.

Let $\mathcal{O}(k, t)$ be the set of nodes at which t originates in k, and $\mathcal{G}(k, t)$ be the set of nodes at which t is gained in k.

Definition 7.3 (Skeleton)**.** A preskeleton k is a *skeleton* if each atom in $unique(k)$ originates in at most one trace, and the node of origination precedes each node that gains the atom, i.e. for every $t \in unique(k)$, $n_0 \in \mathcal{O}(k, t)$ and $n_1 \in \mathcal{G}(k, t)$ implies $n_0 \prec_k n_1$.

Definition 7.4 (Preskeleton Homomorphism)**.** There is a *preskeleton homomorphism* from k_0 to k_1, written $k_0 \xmapsto{\phi,\sigma} k_1$, if ϕ and σ are structure-preserving maps with the following properties:

1. ϕ maps strands of k_0 into those of k_1, and nodes as $\phi((s,p)) = (\phi(s),p)$;
2. σ is a Σ-homomorphism;
3. $n \in \mathit{nodes}(k_0)$ implies $\sigma(\mathit{evt}(\mathit{insts}(k_0),n)) \equiv \mathit{evt}(\mathit{insts}(k_1),\phi(n))$;
4. $n_0 \prec_{k_0} n_1$ implies $\phi(n_0) \prec_{k_1} \phi(n_1)$;
5. $\sigma(\mathit{non}(k_0)) \subseteq \mathit{non}(k_1)$;
6. $\sigma(\mathit{unique}(k_0)) \subseteq \mathit{unique}(k_1)$;
7. $t \in \mathit{unique}(k_0)$ implies $\phi(\mathcal{O}(k_0,t)) \subseteq \mathcal{O}(k_1,\sigma(t))$.

A homomorphism is *strandwise injective* if its strand map is injective. Two preskeletons are isomorphic if they are related by strandwise injective homomorphism in both directions.

Condition 7 ensures that if $t \in \mathit{unique}(k_0)$ originates at (s,p) then when we apply σ to the strand s, $\sigma(t)$ neither originates nor is gained at $(\phi(s),j)$ for $j < p$ ($\sigma(t)$ can never originate or be gained later). In other words, σ alone will either satisfy or violate the last clause.

Definition 7.5 (Homomorphism Composition)**.** Let $k_0 \stackrel{\psi_0}{\longmapsto} k_1 \stackrel{\psi_1}{\longmapsto} k_2$ where $\psi_0 = (\phi_0,\sigma_0)$ and $\psi_1 = (\phi_1,\sigma_1)$. Then the composition of ψ_0 and ψ_1 is defined as $\psi_1 \circ \psi_0 = (\phi_1 \circ \phi_0, \sigma_1 \circ \sigma_0)$ where $k_0 \stackrel{\psi_1\circ\psi_0}{\longmapsto} k_2$.

Since homomorphisms may arbitrarily add some structure such as adding terms to *unique* and *non*, it is possible to have $k_0 \stackrel{\phi,\sigma}{\longmapsto} k_1$ and $k_0 \stackrel{\phi,\sigma}{\longmapsto} k_2$ with $k_1 \neq k_2$. For example k_1 and k_2 could be identical except that $\mathit{non}(k_1) \subsetneq \mathit{non}(k_2)$. We would like to be able to talk about the image of k_0 under (ϕ,σ).

Definition 7.6 (Image of Homomorphism)**.** Preskeleton k_1 is the *image of* k_0 *under* $\psi = (\phi,\sigma)$ written $\psi(k_0) = k_1$ iff

1. $k_0 \stackrel{\phi,\sigma}{\longmapsto} k_1$
2. $\prec_{k_1} = \phi(\prec_{k_0})^*$
3. $\mathit{unique}(k_1) = \sigma(\mathit{unique}(k_0))$
4. $\mathit{non}(k_1) = \sigma(\mathit{non}(k_0))$

5. $ht(k_1, \phi(s)) = \max\{ht(k_0, s') \mid \phi(s') = \phi(s)\}$ for $s < |\, inst(k_0)|$

6. ϕ is surjective

This definition guarantees that the image does not contain extra strands or nodes, and that it only contains ordering relations and origination assumptions necessary for ψ to be a homomorphism. Whenever there is a skeleton k_1 such that $k_0 \stackrel{\psi}{\longmapsto} k_1$, then $\psi(k_0)$ is well-defined, and we can view $\psi(k_0)$ as being included into k_1. That is, $k_0 \stackrel{\psi}{\longmapsto} \psi(k_0) \stackrel{\phi_{\mathrm{id}}, \sigma_{\mathrm{id}}}{\longmapsto} k_1$.

Given $(\phi_{\mathrm{id}}, \sigma)$, it is still possible for $(\phi_{\mathrm{id}}, \sigma)(k_0)$ not to exist because σ may violate the last clause of Definition 7.4. However, if there is some ϕ for which $(\phi, \sigma)(k_0)$ is well-defined, then $(\phi_{\mathrm{id}}, \sigma)(k_0)$ is also well-defined.

Lemma 7.1. If $k_0 \stackrel{\phi,\sigma}{\longmapsto} k_1$, then there is a preskeleton k such that $k_0 \stackrel{\phi_{id},\sigma}{\longmapsto} k \stackrel{\phi,\sigma_{id}}{\longmapsto} k_1$.

Proof. Let $k = (\phi_{\mathrm{id}}, \sigma)(k_0)$. Then the homomorphism $k \stackrel{\phi,\sigma_{id}}{\longmapsto} k_1$ is well-defined which the reader can easily check by verifying each of the seven clauses of Definition 7.4. □

Corollary 1. If $k_0 \stackrel{\phi,\sigma_{id}}{\longmapsto} k_1 \stackrel{\phi_{id},\sigma}{\longmapsto} k_2$, then there is a preskeleton k such that $k_0 \stackrel{\phi_{id},\sigma}{\longmapsto} k \stackrel{\phi,\sigma_{id}}{\longmapsto} k_2$.

Proof. We can compose the two homomorphisms to see that $k_0 \stackrel{\phi,\sigma}{\longmapsto} k_2$. We then apply the previous lemma to find k. □

The following definition is frivolous as it is not used anywhere. We're just trying out definitions to explore links to category theory.

Definition 7.7 (CPSA Category)**.** Let $\mathfrak{C}(X)$ be a CPSA algebra generated by X (see Definition 3.1), and $\mathfrak{K}(X)$ be a free algebra defined by the carrier set for sort preskel in $\mathfrak{C}(X)$. In a CPSA category

1. $\mathfrak{K}(X)$ is an object, for each variable set X,

2. the set of arrows is $(\mathbb{N} \to \mathbb{N}) \times (\mathfrak{A}_\top(X) \to \mathfrak{A}_\top(Y))$, where the second component is a homomorphism of the message algebra,

3. Definition 7.4 defines the domain and the codomain of each arrow,

4. component function composition defines arrow composition, and

5. component identity functions define the arrow identity.

Definition 7.8 (Pruned Skeleton)**.** A skeleton k has a set of *redundant strands* S if there is a substitution σ that is a variable renaming, a strand mapping ϕ such that $s \in S$ implies there is some $s' \notin S$ such that $\phi(s) = \phi(s')$ and the height of s is no greater than the height of s', a skeleton k' such that $k \stackrel{\phi,\sigma}{\longmapsto} k'$ where $k' = (\phi, \sigma)(k)$, and a homomorphism $k' \stackrel{\phi',\sigma'}{\longmapsto} k$ such that $\sigma \circ \sigma' = \sigma_{\mathrm{id}}$ and $\phi \circ \phi' = \phi_{\mathrm{id}}$. A skeleton k is *pruned* if it contains no redundant strands.

Suggestion from Carolyn Talcott: Define pruned skeleton as the result of deleting redundant set of strands. Then prove that we can use homomorphisms with the right properties to perform this 'deletion'.

The concept of an execution skeleton is introduced to relate a skeleton that contains only regular strands with its executions. An execution skeleton may include penetrator strands.

Definition 7.9 (Used)**.** Message t_0 is *used in* t_1 if t_0 or $inv(t_0)$ is accessible in t_1 and t_0 is not carried by t_1.

The accessibility of a term is defined in Definition 2.9.

Definition 7.10 (Execution Skeleton)**.** The *execution skeleton* k of bundle $\mathcal{B}$ over instances I has the following properties.

1. $insts(k) = I$.
2. $\prec_k = \prec$.
3. $unique(k)$ is the set of atoms that originate uniquely in $\mathcal{B}$.
4. $non(k)$ is the set of atoms used in the traces of I.

Definition 7.11 (Skeleton Compatible Executions)**.** A bundle $\mathcal{B}$ is *compatible* with skeleton k if there is a homomorphism from k to the execution skeleton of $\mathcal{B}$.

Definition 7.12 (Realized Skeleton)**.** A bundle $\mathcal{B}$ *realizes* skeleton k if $\mathcal{B}$ is compatible with k, and the structure preserving map (ϕ, σ) has the property that σ is a bijection, ϕ is a bijection between the strands in k and the regular strands in the bundle's execution skeleton, and ϕ preserves the height of the strands it maps.

A bundle models a realized skeleton if it realizes it. A bundle $\mathcal{B}$ models a preskeleton k_0 if there is a realized skeleton k_1 such that $k_0 \stackrel{\phi,\sigma}{\longmapsto} k_1$ and $\mathcal{B}$ models k_1.

Discussion

In the refinement of strand space theory, there is no global strand space on which all analysis is based. Instead, each skeleton and execution defines its own strand space, and homomorphisms establish relations between them. The definition of a protocol depends on no strand space.

The definitions in this section are the obvious refinements that result from using finite ordered strand spaces. In the CPSA program, pruned skeletons are used for skeletons in the implementation of the CPSA algorithm in [2]. In [2], skeletons were not required to respect origination, but that was just an oversight.

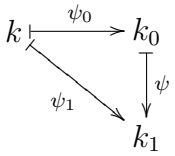
8 Reductions on Preskeletons

This section describes the algorithm used to transform a preskeleton into a skeleton as a setwise term reduction system $(\mathfrak{K}, \twoheadrightarrow)$. Recall that the relation $k \rightsquigarrow K$ in Section 4 is defined in terms of $\rightarrow \subseteq \mathfrak{K} \times \mathfrak{K}$ by specifying $\{k\} \twoheadrightarrow K$ using $\rightarrow$.

If a preskeleton k is not a skeleton, then it is either because some $t \in \mathit{unique}(k)$ actually originates at more than one node, or because for some $t \in \mathit{unique}(k)$, there is a node $n_1 \in \mathcal{G}(k,t)$, and a node $n_0 \in \mathcal{O}(k,t)$ such that $n_0 \not\prec_k n_1$. These obstructions are resolved via identifying strands and enriching node orderings respectively. We show that we can always resolve the first obstruction before resolving the second obstruction. Moreover, if the first obstruction is resolvable, then there is a canonical resolution (although there may be non-canonical choices to reach it). We call this canonical resolution a pre-hull. Then if the second obstruction is also resolvable, it also has a canonical resolution. This canonical resolution is a skeleton which we call a hull. We begin by giving the definitions of pre-hull and of hull.

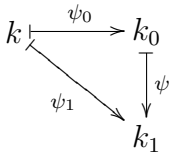
Definition 8.1. Given a preskeleton k, a *pre-hull* of k is a hulled preskeleton k_0 together with a homomorphism $k \stackrel{\psi_0}{\longmapsto} k_0$ such that for any homomorphism

$k \overset{\psi_1}{\longmapsto} k_1$ to a hulled preskeleton, there is a unique homomorphism $k_0 \overset{\psi}{\longmapsto} k_1$ such that $\psi_1 = \psi \circ \psi_0$



A preskeleton k may not have a pre-hull, but if it does, then the definition implies that it is unique up to isomorphism. Also, every hulled preskeleton is its own pre-hull where ψ_0 is the identity homomorphism.

Definition 8.2. Given a preskeleton k, a *hull* of k is a skeleton k_0 together with a homomorphism $k \overset{\psi_0}{\longmapsto} k_0$ such that for any homomorphism $k \overset{\psi_1}{\longmapsto} k_1$ to a skeleton, there is a unique homomorphism $k_0 \overset{\psi}{\longmapsto} k_1$ such that $\psi_1 = \psi \circ \psi_0$



Just like pre-hulls, a preskeleton may not have a hull, but if it does, then it is unique up to isomorphism. Also, every skeleton is its own hull where ψ_0 is the identity homomorphism.

We next show how to take advantage of unification in the Σ-algebra to provide a sort of unification of strands. Before unifying two strands, we will unify their traces.

Definition 8.3 (Trace Unification)**.** Let k be a preskeleton which contains strands s and s'. We say that substitution σ *unifies the messages of* s *and* s' iff $\sigma(evt(k,(s,p))) \equiv \sigma(evt(k,(s',p)))$ for every $p < height(s)$. We say that homomorphism $k \overset{\phi,\sigma}{\longmapsto} k_0$ *unifies the traces of* s *and* s' iff σ unifies the messages of s and s'. We say that the traces of s and s' are *unifiable* if there is a (ϕ, σ) which unifies them.

Thus, if (ϕ, σ) unifies the traces of s and s', not only does σ unify the messages but also $t \in unique(k_0)$ implies $\phi(\mathcal{O}(k_0, t)) \subseteq \mathcal{O}(k_1, \sigma(t))$ by Clause 7 in the definition of a homomorphism (Definition 7.4). In fact, as we already saw, it is generally possible for σ to unify the messages, without respecting

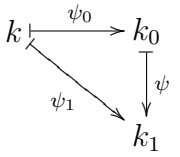
Clause 7. In this case, σ unifies the messages of s and s', but the traces of s and s' are not unifiable. By pairing σ with ϕ, trace unification only allows homomorphisms with substitutions σ which do not violate Clause 7.

By recalling the discussion which follows Definition 7.6, we can see that if (ϕ, σ) unifies the traces of s and s', then so does $(\phi_{\mathrm{id}}, \sigma)$. Furthermore, if $(\phi_{\mathrm{id}}, \sigma)$ unifies the traces of s and s', then so does $(\phi_{\mathrm{id}}, \sigma')$ where σ' unifies the messages of s and s', and $\sigma' \preceq \sigma$. We can view this another way. Let σ be the most general unifier of the messages of s and s'. If $(\phi_{\mathrm{id}}, \sigma)$ does not unify the traces, then the traces are not unifiable.

Definition 8.4 (Substitution Reduction)**.** Preskeleton k_0 reduces to preskeleton k_1 by the substitution σ, written $k_0 \xrightarrow{\mathbb{S}_\sigma} k_1$, iff $k_1 = \psi(k_0)$ where $\psi = (\phi_{\mathrm{id}}, \sigma)$.

The next lemma states that if k is a skeleton in which the traces of two strands s and s' are unifiable, then there is a kind of most general trace unifier. Since this most general trace unifier will turn out to always have the form $(\phi_{\mathrm{id}}, \sigma)$, we can use the corresponding $\mathbb{S}_\sigma$ as a way of implementing this trace unification.

Lemma 8.1. Let k be a preskeleton which contains strands s and s'. Suppose the traces of s and s' are unifiable. Then there is a preskeleton k_0 and a homomorphism $k \xmapsto{\psi_0} k_0$ which unifies the traces of s and s' such that for every homomorphism $k \xmapsto{\psi_1} k_1$ which unifies the traces of s and s', there is a unique homomorphism $k_0 \xmapsto{\psi} k_1$ so that $\psi_1 = \psi \circ \psi_0$.



Proof. Since the traces of s and s' are unifiable by some $(\hat{\phi}, \hat{\sigma})$, we know that $(\phi_{\mathrm{id}}, \hat{\sigma})$ also unifies the traces. Furthermore, there is a most general unifier σ_0 which unifies the messages of s and s'. Since $\sigma_0 \preceq \hat{\sigma}$ we know that $(\phi_{\mathrm{id}}, \sigma_0)$ also unifies the traces. Let $\psi_0 = (\phi_{\mathrm{id}}, \sigma_0)$ and let $k_0 = \psi_0(k)$. We show that this is the preskeleton and homomorphism we want.

The homomorphism ψ_0 unifies the traces of s and s' by construction. Now let $k \xmapsto{\psi_1} k_1$ with $\psi_1 = (\phi_1, \sigma_1)$, be a homomorphism which unifies the traces

of s and s'. Since σ_0 is the most general unifier of the messages of s and s', there is a unique σ such that $\sigma_1 = \sigma \circ \sigma_0$. Then by letting $\psi = (\phi_1, \sigma)$ we see that $\psi \circ \psi_0 = (\phi_1, \sigma) \circ (\phi_{id}, \sigma_0) = (\phi_1, \sigma \circ \sigma_0) = (\phi_1, \sigma_1) = \psi_1$. It is clear that ϕ_1 is the unique node map that will work, and we already noted that σ must also be unique. □

We can find the k_0 of Lemma 8.1 (when it exists) by finding the most general σ_0 which simultaneously unifies the corresponding messages in the traces of s and s' and applying the reduction $\mathbb{S}_{\sigma_0}$ to k. This reduction will fail if the traces of s and s' are not unifiable, and it will succeed otherwise. Furthermore, Lemma 8.1 guarantees that there is at most a single k_0 such that $k \xrightarrow{\mathbb{S}_{\sigma_0}} k_0$ where σ_0 is the most general unifier of the messages of strands s and s'.

Definition 8.5 (Strand Unification)**.** Let k be a preskeleton with strands s and s'. We say that a homomorphism $\psi = (\phi, \sigma)$ *unifies the strands* s and s' iff $\phi(s) = \phi(s')$. We say the strands s and s' are *unifiable* if there is a homomorphism which unifies them.

Note that if ψ unifies two strands, then ψ unifies their traces, but the converse is not true in general. Thus, unification of two strands can be broken into two steps. First we unify their traces, then we unify the strands in a very simple way. We already saw that the first step can be performed in a most general way, as codified by Lemma 8.1. We now want to provide a similar notion for the second step.

Suppose the trace of strand s is a prefix of the trace of strand s' in preskeleton k_0. In other words, $(\phi_{\mathrm{id}}, \sigma_{\mathrm{id}})$ unifies the traces of s and s'. This would be the case, for example, if k_0 was the result of unifying the traces of s and s' in some other preskeleton. In that case, there is a ψ which unifies the strands s and s', namely $\psi = (\phi_{s,s'}, \sigma_{\mathrm{id}})$ where

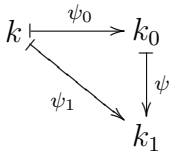
$$\phi_{s,s'}(j) = \begin{cases} \phi_s(s') & \text{if } j = s \\ \phi_s(j) & \text{otherwise} \end{cases}$$

$$\phi_s(j) = \begin{cases} j - 1 & \text{if } j > s \\ j & \text{otherwise.} \end{cases}$$

Definition 8.6 (Compression Reduction)**.** Preskeleton k_0 reduces to preskeleton k_1 by compressing the strand s into s', written $k_0 \xrightarrow{\mathbb{C}_{s,s'}} k_1$, iff $\psi(k_0) = k_1$, with $\psi = (\phi_{s,s'}, \sigma_{\mathrm{id}})$.

The compression reduction $\mathbb{C}_{s,s'}$ can only be performed on k_0 if $(\phi_{\text{id}}, \sigma_{\text{id}})$ unifies the traces of s and s', because otherwise $(\phi_{s,s'}, \sigma_{\text{id}})(k_0)$ is not well-defined. We generally use this reduction when unifying strands after having unified their traces. This reduction is used to implement a kind of most general unification of two strands once their traces have been unified.

Lemma 8.2. Let k be a preskeleton in which $(\phi_{\text{id}}, \sigma_{\text{id}})$ unifies the traces of two strands s and s', and suppose the strands s and s' are unifiable. Then there is a preskeleton k_0 and a homomorphism $k \xmapsto{\psi_0} k_0$ which unifies the strands s and s' such that for every homomorphism $k \xmapsto{\psi_0} k_0$ which unifies the strands s and s', there is a unique homomorphism $k_0 \xmapsto{\psi} k_1$ so that $\psi_1 = \psi \circ \psi_0$.



Proof. Let $\psi_0 = (\phi_{s,s'}, \sigma_{id})$, and $\psi_0(k) = k_0$. We will show that this is the preskeleton and homomorphism we want.

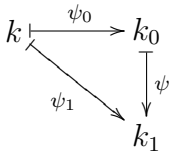
By the definition of $\phi_{s,s'}$ we see that ψ_0 unifies the strands s and s'. Let $k \xmapsto{\psi_1} k_1$ with $\psi_1 = (\phi_1, \sigma_1)$ be any homomorphism which unifies the strands s and s'. We let $\psi = (\phi, \sigma_1)$ where ϕ is a node map we still need to define. Since $\phi_{s,s'}$ maps onto the strands of k_0, ϕ is completely determined by where it sends $\phi_{s,s'}(j)$. But in order for $\psi_1 = \psi \circ \psi_0$ we are forced to define $\phi(\phi_{s,s'}(j)) = \phi_1(j)$. We can do this because $\phi_{s,s'}$ does not identify any strands except for s and s', and because ϕ_1 also identifies s and s'. Using this ϕ, it is clear that $\psi_1 = \psi \circ \psi_0$. To see that ψ is unique, we note that we had no choice for either the substitution σ_1 or the node map ϕ. □

We can find the k_0 of Lemma 8.2 (when it exists) by applying $\mathbb{C}_{s,s'}$ to k. This reduction will fail if the strands s and s' are not unifiable or if the trace of s is not a prefix of the trace of s', and it will succeed otherwise. Moreover, Lemma 8.2 guarantees that there is at most a single k_0 such that $k \xrightarrow{\mathbb{C}_{s,s'}} k_0$.

The substitution and compression reductions are used to unify two strands in two separate steps. When two strands are unifiable, then their traces are also unifiable. We may thus use Lemma 8.1 to unify their traces in a most general way. This will cause the trace of one of the strands to be a prefix

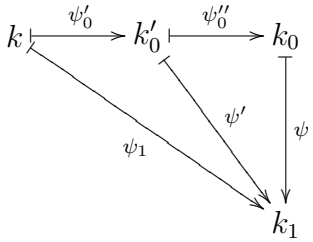
of the other, thereby enabling compression. The resulting preskeleton is the most general one which unifies the two strands, as we show in the following lemma.

Lemma 8.3. Let k be a preskeleton containing strands s and s' which are unifiable. Then there is a preskeleton k_0 and a homomorphism $k \overset{\psi_0}{\longmapsto} k_0$ which unifies s and s' such that for every homomorphism $k \overset{\psi_1}{\longmapsto} k_1$ which unifies s and s', there is a unique homomorphism $k_0 \overset{\psi}{\longmapsto} k_1$ so that $\psi_1 = \psi \circ \psi_0$.



Proof. Since the strands s and s' are unifiable, their traces are also unifiable. Thus, by Lemma 8.1, we can find a preskeleton k'_0 and a homomorphism $k \overset{\psi'_0}{\longmapsto} k'_0$ so that any homomorphism from k which unifies the traces of s and s' factors through ψ'_0. The preskeleton k'_0 now satisfies the hypotheses of Lemma 8.2, so we can find a preskeleton k''_0 and a homomorphism $k'_0 \overset{\psi''_0}{\longmapsto} k''_0$ so that any homomorphism from k'_0 which unifies s and s' factors through ψ''_0. Let $\psi_0 = \psi''_0 \circ \psi'_0 = (\phi_{s,s'}, \sigma)$ where σ is the mgu of the traces of s and s', and let $k_0 = k''_0$. It remains to show that this ψ_0 and k_0 have the desired properties.

First, it is clear that ψ_0 unifies s and s'. Now let $k \overset{\psi_1}{\longmapsto} k_1$ be an arbitrary homomorphism which unifies s and s'. By Lemma 8.1, ψ_1 factors uniquely through ψ'_0 as $\psi' \circ \psi'_0$. Moreover, ψ' is a homomorphism from k'_0 which unifies s and s', thus by Lemma 8.2, ψ' factors uniquely through ψ''_0 as $\psi' = \psi'' \circ \psi''_0$. Thus ψ_1 factors uniquely through $\psi_0 = \psi''_0 \circ \psi'_0$ as desired.

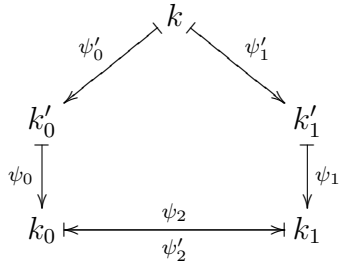


□

We thus now have a canonical way of unifying two strands when they are unifiable.

Lemma 8.4. Let k be a preskeleton containing strands s_0, s_0', s_1, s_1' (not necessarily all distinct) such that the pairs of strands (s_0, s_0') and (s_1, s_1') are simultaneously unifiable. Let k_0 be the result of first unifying s_0 and s_0' and then s_1 and s_1' according to Lemma 8.3. Let k_1 be the result of first unifying s_1 and s_1' and then s_0 and s_0' also according to Lemma 8.3. Then k_0 and k_1 are isomorphic.

Proof. We can repeatedly apply Lemma 8.3 to obtain the following diagram.



The preskeleton k_0' is the result of unifying strands s_0 and s_0' of k in a most general way, and k_0 is the result of unifying strands $\psi_0'(s_1)$ and $\psi_0'(s_1')$ of k_0' in a most general way. Similarly, k_1' is the result of unifying strands s_1 and s_1' of k in a most general way, and k_1 is the result of unifying strands $\psi_1'(s_0)$ and $\psi_1'(s_0')$ of k_1' in a most general way.

We can infer the homomorphism $k_0 \overset{\psi_2}{\longmapsto} k_1$ and $k_1 \overset{\psi_2'}{\longmapsto} k_0$ from Lemma 8.3 as the unique homomorphisms which make the diagram commute. By the uniqueness of these arrows we can infer that $\psi_2 \circ \psi_2' = id_{k_1}$ and that $\psi_2' \circ \psi_2 = id_{k_0}$. Thus ψ_2 and ψ_2' are isomorphisms and they are inverses of each other. Thus k_0 and k_1 are isomorphic. □

Lemma 8.4 implies that if we want to simultaneously unify several sets of strands which are simultaneously unifiable, any order of pairwise strand unification will result in a skeleton which is isomorphic to the result of every other order. However, given a preskeleton k which is not a hulled preskeleton, we do not want to unify *every* pair of strands which are unifiable. We only want to unify those pairs of strands which demonstrate that for some $t \in unique(k)$, $\mathcal{O}(k, t)$ has more than one node. For this purpose we use another reduction which is built out of $\mathbb{S}_\sigma$ and $\mathbb{C}_{s,s'}$.

Recall that the relation $k \rightsquigarrow K$ in Section 4 is defined in terms of $\rightarrow \subseteq \mathfrak{K} \times \mathfrak{K}$ by specifying $\{k\} \twoheadrightarrow K$ using $\rightarrow$.

Definition 8.7 (Hulling Reduction)**.** Preskeleton k_0 reduces to preskeleton k_1 by hulling strands s and s', written $k_0 \xrightarrow{\mathbb{H}_{s,s'}} k_1$, iff there is some $t \in \mathit{unique}(k_0)$ and there are distinct strands s and s' such that $\{(s,p),(s',p')\} \subseteq \mathcal{O}(k_0,t)$, and the strands s and s' are unifiable, and k_1 is the preskeleton guaranteed by Lemma 8.3. For the setwise hulling relation, $\{k_0\} \xrightarrow{\mathbb{H}_{s,s'}} \{k_1 \mid k_0 \xrightarrow{\mathbb{H}_{s,s'}} k_1\}$, when k_0 has a message in $\mathit{unique}(k_0)$ that originates on both s and s'.

This hulling reduction $\mathbb{H}_{s,s'}$ implements the "most general strand unifier" ψ_0 from Lemma 8.3 when it exists, but it is only applicable when s and s' originate the same $t \in \mathit{unique}(k)$. By the proof of Lemma 8.3, $k_0 \xrightarrow{\mathbb{H}_{s,s'}} k_1$, iff there is a (unique) preskeleton k such that $k_0 \xrightarrow{\mathbb{S}_\sigma} k \xrightarrow{\mathbb{C}_{s,s'}} k_1$, where σ is the most general unifier of the messages of s and s'.

Given a preskeleton k which is not a hulled preskeleton, it is possible that simultaneously unifying all pairs of strands s and s' which both originate some $t \in \mathit{unique}(k)$ will not produce a hulled preskeleton. The unification process may introduce more points of origination of terms which are meant to be uniquely originating. However, the next lemma shows that repeatedly resolving these obstructions will yield a pre-hull.

Lemma 8.5. Suppose that there is a homomorphism from k into a hulled preskeleton. Then k has a pre-hull.

Proof. If k is not a hulled preskeleton, then it has (possibly several) pairs of strands s and s' which each originate the same term $t \in \mathit{unique}(k)$. All of these pairs are simultaneously unifiable since the homomorphism from k into a hulled preskeleton which is assumed to exist performs that unification. Therefore by Lemmas 8.3 and 8.4, we may unify these pairs s and s' in any order in a most general way to find a homomorphism $k \xmapsto{\psi_0} k_0$ which simultaneously unifies all the pairs s and s', such that for any homomorphism $k \xmapsto{\psi_1} k_1$ which unifies these pairs there is a unique $k_0 \xmapsto{\psi} k_1$ such that $\psi_1 = \psi \circ \psi_0$.

By the properties of ψ_0 and k_0, if k_0 is a hulled preskeleton it is the pre-hull of k. If not, then k_0 again satisfies the hypothesis of this lemma,

so we can repeat the process of the above paragraph. Since each iteration reduces the number of strands, this process will eventually terminate. By the properties of the resulting preskeleton at each step, it will terminate with the pre-hull. □

Lemma 8.6. If a preskeleton k has a pre-hull, then repeated applications of $\mathbb{H}_{s,s'}$ will terminate in the pre-hull of k.

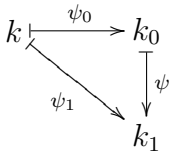
Proof. For every pair of strands s and s' which are eventually unified in the pre-hull, there is a sequence of hulling reductions which causes s and s' to originate the same term which should be uniquely originating. Furthermore, by Lemma 8.4 this obstruction is introduced by the end of every reordering of such a sequence, although it may be introduced earlier. Thus, we may apply the hulling reductions in any order, and we will eventually unify the same sets of strands. Again by Lemma 8.4, the preskeletons resulting from each order of hulling reductions will all be isomorphic. Since Lemma 8.5 shows that at least one such order results in the pre-hull, every order will result in the pre-hull. □

Definition 8.8 (Order Enrichment)**.** Suppose hulled preskeleton k_0 is not a skeleton. Hulled preskeleton k_0 reduces to skeleton k_1 by order enrichment, written $k_0 \xrightarrow{\mathbb{O}} k_1$, iff k_1 is the result of adding node orderings implied by origination. That is,

1. $\prec_{k_1} = (\prec_{k_0} \cup \{n_0, n_1 \mid n_0 \in \mathcal{O}(k_0, t) \wedge n_1 \in \mathcal{G}(k_0, t)\})^*$,
2. $ht(k_1, s) = ht(k_0, s)$ for $s < |\, inst(k_0)|$,
3. $unique(k_1) = unique(k_0)$, and
4. $non(k_1) = non(k_0)$.

There is a homomorphism from k_0 to k_1 that is an embedding. For the setwise order enrichment reduction, $\{k_0\} \xrightarrow{\mathbb{O}}\!\!\!\!\rightarrow \{k_1 \mid k_0 \xrightarrow{\mathbb{O}} k_1\}$ when k_0 is a hulled preskeleton that is not a skeleton.

Lemma 8.7. Suppose k is a preskeleton such that every t in $unique(k)$ originates in k at most once. Suppose also that there is a homomorphism from k to a skeleton. Then there is a skeleton k_0 and a homomorphism $k \xmapsto{\psi_0} k_0$ such that for every homomorphism $k \xmapsto{\psi_1} k_1$ to a skeleton k_1, there is a unique homomorphism $k_0 \xmapsto{\psi} k_1$ so that $\psi_1 = \psi \circ \psi_0$.



Proof. Let k_0 be obtained from k by adding edges to the node ordering relation. That is, $n_0 \prec_{k_0} n_1$ is the transitive closure of pairs such that either $n_0 \prec_k n_1$ or for some $t \in \mathit{unique}(k)$, t originates at n_0 and $n_1 \in \mathcal{G}(k,t)$. Let ψ_0 be the obvious embedding of k into k_0. We must show that k_0 is a skeleton. It will be sufficient to check that we did not introduce any cycles into the node ordering. But, in fact, if $\prec_{k_0}$ had a cycle, then so would the

First, we chose k_0 to be a skeleton. Now let $k \xmapsto{\psi_1} k_1$ be an arbitrary homomorphism to a skeleton k_1. Then we can also apply ψ_1 to k_0, but we must be careful to check that it remains a homomorphism. Only clause 4 of Definition 7.4 could potentially fail. However, assume □

Our current understanding of pruning is this. We have an example that shows that pruning one strand at a time does not remove all redundant strands. We are searching for efficient ways to perform multiple strand pruning in one step, but haven't found one yet.

We also do not know if single strand pruning is confluent.

Definition 8.9 (Pruning)**.** Suppose skeleton k_0 has a redundant strand s. Then there exists a distinct strand s' that describes more specific behavior. Skeleton k_0 reduces to skeleton k_1 by pruning, written $k_0 \xrightarrow{\mathbb{P}_s} k_1$, iff there is a substitution σ such that $\sigma(\mathit{evt}(k_0,(s,p))) \equiv \mathit{evt}(k_0,(s',p))$ for all $p < h$, where h is the height of strand s in k_0, no variable in $\mathit{Dom}(\sigma)$ occurs in the trace of any strand other than s, $t \in \mathit{unique}(k_0)$ implies $\sigma(t) \in \mathit{unique}(k_0)$, $t \in \mathit{non}(k_0)$ implies $\sigma(t) \in \mathit{non}(k_0)$, there is a k such that $k_0 \xrightarrow{\mathbb{S}_\sigma} k \xrightarrow{\mathbb{C}_{s,s'}} k_1$, and if $(s,p) \prec_{k_0} (s'',p'')$ then $(s',p') \prec_{k_0} (s'',p'')$, and if $(s'',p'') \prec_{k_0} (s,p)$ then $(s'',p'') \prec_{k_0} (s',p')$. For the setwise pruning reduction, $\{k_0\} \xrightarrow{\mathbb{P}_s} \{k_1 \mid k_0 \xrightarrow{\mathbb{P}_s} k_1\}$, when there is a k_1 such that $k_0 \xrightarrow{\mathbb{P}_s} k_1$.

Lemma 8.8 (Pruning)**.** Let k be a skeleton with redundant strand s. For every pruned skeleton k_1 such that $k \xmapsto{\psi_1} k_1$, there is a skeleton k_0 and homomorphisms ψ_0 and ψ with $k \xrightarrow{\mathbb{P}_s} k_0$, $k \xmapsto{\psi_0} k_0$, $\psi_1 = \psi \circ \psi_0$, and $\psi_0 = (\phi_{s,s'}, \sigma)$, where strand s' and substitution σ are as specified in Definition 8.9.

Proof. Pruning operations commute. Suppose k has two redundant strands, s and s'. If s describes more specific behavior and is used to justify pruning s', then the strand that justifies the pruning of s can serve the same purpose.
The proof is not complete. Something more needs to be added here. □

Notice that a setwise hulling reduction may produce the empty set, but a setwise order enrichment and pruning reduction never does.

Let reduction $\twoheadrightarrow = \bigcup_{s,s'} \xrightarrow{\mathbb{H}_{s,s'}}\!\!\!\!\!\!\twoheadrightarrow \cup \xrightarrow{\mathbb{O}}\!\!\!\!\!\!\twoheadrightarrow \cup \bigcup_s \xrightarrow{\mathbb{P}_s}\!\!\!\!\!\!\twoheadrightarrow$.

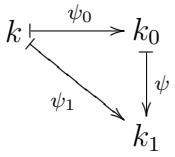
Theorem 8.9. The reduction $\twoheadrightarrow$ is convergent.

Proof. The reduction $\twoheadrightarrow$ is confluent by Lemmas 8.3 and 8.8. It's convergent because the number of hulling and pruning steps is bounded by the number of strands in a preskeleton. □

Definition 8.10 (Preskeleton Reduction System)**.** Preskeleton k_0 reduces to pruned skeleton k_1, written $k_0 \xrightarrow{skel} k_1$, if $\{k_0\} \twoheadrightarrow^* K$, $k_1 \in K$, and K is a normal form of $\twoheadrightarrow$.

It is easy to show $k_0 \xrightarrow{skel} k_1$ implies $k_0 \mapsto k_1$. Furthermore, the structure-preserving map that demonstrates the homomorphism is easy to derive. For each pruned skeleton k, $k \xrightarrow{skel} k$.

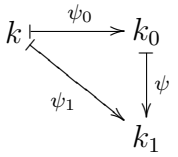
Theorem 8.10 (Preskeleton Reduction System Correct)**.** Let k be a preskeleton. For every pruned skeleton k_1 such that $k \xmapsto{\psi_1} k_1$, there is a pruned skeleton k_0 and homomorphisms ψ_0 and ψ with $k \xrightarrow{skel} k_0$, $k \xmapsto{\psi_0} k_0$, and $\psi_1 = \psi \circ \psi_0$.



Old stuff.

Lemma 8.11 (Hulling)**.** Suppose $k \mapsto k'$ with k a preskeleton and k' a hulled preskeleton. There exists a set of homomorphisms Ψ and a set of hulled preskeletons K, such that for every hulled preskeleton k_1 and every homomorphism $k \xmapsto{\psi_1} k_1$, for some $\psi_0 \in \Psi$, $k_0 \in K$, and ψ, $\psi_1 = \psi \circ \psi_0$, ψ is unique to within isomorphism, and k_0 is isomorphic to some k_2 with $k \rightarrow^* k_2$, where reduction $\rightarrow = \bigcup_{s,s'} \xrightarrow{\mathbb{H}_{s,s'}}$.

Theorem 8.12. Suppose $k \mapsto k'$ with k a preskeleton and k' a pruned skeleton. There exists a set of homomorphisms Ψ and a set of pruned skeletons K, such that for every pruned skeleton k_1 and every homomorphism $k \xmapsto{\psi_1} k_1$, for some $\psi_0 \in \Psi$, $k_0 \in K$, and ψ, $\psi_1 = \psi \circ \psi_0$, ψ is unique to within isomorphism, and k_0 is isomorphic to some k_2 with $k \xrightarrow{skel} k_2$.



9 Penetrator Derivable

For each algebra, the powers of the adversary are defined by a set of roles. For the Basic Crypto Signature in Figure 1, the traces of the penetrator roles are in Figure 5. Penetrator roles make no origination assumptions.

The context in which penetrator strands appear determine the messages the adversary can derive. The context includes previously sent messages and atoms it is forbidden to originate. An atom that is assumed to be non-originating must be avoided as is a uniquely originating atom that is assumed to originate on a regular strand.

The ternary relation $T_p : T_a \vdash t$ states that message t is penetrator derivable from previously sent messages T_p while avoiding atoms T_a. The relation is defined by a set of inference rules. Most of the rules are justified by a penetrator role that when instantiated, derives a message in the conclusion of the rule.

The first rule states that no additional penetrator behavior is required to derive t if it has been previously sent.

$$\frac{t \in T_p}{T_p : T_a \vdash t}$$

A uniquely originating atom need not be avoided if it has been sent.

$$\frac{T_p : T_a \vdash t}{\{t_0\} \cup T_p : \{t_0\} \cup T_a \vdash t} \tag{1}$$

There are two decomposition steps available to the penetrator.

$$\frac{\{t_0, t_1\} \cup T_p : T_a \vdash t}{\{(t_0, t_1)\} \cup T_p : T_a \vdash t} \quad [\text{by } sep(t_0, t_1)] \tag{2}$$

$$\frac{T_p : T_a \vdash inv(t_1) \qquad \{t_0, \{\!|t_0|\!\}_{t_1}\} \cup T_p : T_a \vdash t}{\{\{\!|t_0|\!\}_{t_1}\} \cup T_p : T_a \vdash t} \quad [\text{by } dec(t_0, t_1)] \qquad (3)$$

There are two constructive steps.

$$\frac{T_p : T_a \vdash t_0 \qquad T_p : T_a \vdash t_1}{T_p : T_a \vdash (t_0, t_1)} \quad [\text{by } cat(t_0, t_1)]$$

$$\frac{T_p : T_a \vdash t_0 \qquad T_p : T_a \vdash t_1}{T_p : T_a \vdash \{\!|t_0|\!\}_{t_1}} \quad [\text{by } enc(t_0, t_1)]$$

There are three rules for indivisible messages.

$$T_p : T_a \vdash C_i \quad [\text{by } tag(C_i)]$$

$$\frac{t \notin T_a \qquad t \text{ an atom}}{T_p : T_a \vdash t} \quad [\text{by } base(t)]$$

A non-base sorted variable is derivable in a bundle that instantiates it with any message other than an element of $X_\top$.

$$\frac{t \in X_\top}{T_p : T_a \vdash t}$$

Definition 9.1 (Outbound predecessors)**.** The *outbound predecessors* of skeleton k at n is $outpred(k, n) = \{msg(k, n_0) \mid n_0 \prec_k n, n_0 \text{ is transmitting}\}$.

Definition 9.2 (Avoidance Set)**.** The *avoidance set* of skeleton k is $avoid(k) = non(k) \cup \{t \mid t \in unique(k) \wedge |\mathcal{O}(k, t)| = 1\}$.

An atom in $avoid(k)$ is not available to the penetrator, except if it is exposed by a messages transmission. Clearly, only uniquely originating atoms can be exposed.

Definition 9.3 (Derivable Before)**.** A message t is *derivable before* reception node n in skeleton k, written $der(k, n, t)$, if $T_p : T_a \vdash t$ where $T_p = outpred(k, n)$ and $T_a = avoid(k)$.

Definition 9.4 (Realized Node)**.** A reception node n is *realized* in skeleton k if $msg(k, n)$ is derivable before n in k.

Notice that one can read off penetrator behavior from the proof tree used to demonstrate that $msg(k, n)$ is derivable before n in k. For example, if a decryption step is required by the proof, an instance of the penetrator's decryption role is indicated. In a bundle, for a non-base sorted variable, there is a substitution that maps the variable to a message that is not a non-base sorted variable. The substitution determines the penetrator behavior associated with the variable.

Theorem 9.1 (Realized Skeleton)**.** A skeleton is realized if and only if all of its reception nodes are realized.

Proof. Given a skeleton k in which all of its reception nodes are realized, the combination of the regular behavior in the skeleton, the penetrator behavior specified by the proof trees used to demonstrate each node is realized, and a substitution for non-base sorted variables determines a bundle. The skeleton of the bundle may have more non-originating atoms than is in $non(k)$, however since the extra non-originating atoms are derivable by the bundle that realizes k, the proof trees for those atoms specify any additional penetrator behavior required.
I haven't figure out how to do the "only if" part of this proof yet. □

9.1 Implementation

The derivable before a node predicate is implemented using auxiliary functions.

Definition 9.5 (Buildable)**.** Message t is *buildable* from previously sent messages T_p while avoiding T_a, written $bld(t, T_p, T_a)$, if $T_p : T_a \vdash t$ without the use of Inference Rules 1, 2, and 3.

Consider the following reduction system based on Inference Rules 1, 2, and 3.

$$\begin{array}{rll} \{t\} \cup T_p : T_a \rightarrow T_p : T_a \setminus \{t\} & & \text{if } t \text{ is an atom or in } X_\top \\ \{(t_0, t_1)\} \cup T_p : T_a \rightarrow \{t_0, t_1\} \cup T_p : T_a & & \\ \{\{|t_0|\}_{t_1}\} \cup T_p : T_a \rightarrow \{t_0, \{|t_0|\}_{t_1}\} \cup T_p : T_a & & \text{if } t_0 \notin T_p \text{ and} \\ & & bld(inv(t_1), T_p, T_a) \end{array}$$

Definition 9.6 (Decompose)**.** Previously sent messages T_p and avoidance set T_a *decompose* to T'_p, T'_a, written $decompose(T_p, T_a) = (T'_p, T'_a)$, if $T_p : T_a \rightarrow^* T'_p : T'_a$ and (T'_p, T'_a) is a normal form of reduction $\rightarrow$.

The penetrator derivable predicate $T_p : T_a \vdash t$ is implemented as

$$\begin{array}{l} T_p : T_a \vdash t = \\ \quad \textbf{let } T_p', T_a' = \mathit{decompose}(T_p, T_a) \textbf{ in} \\ \quad \mathit{bld}(t, T_p', T_a') \end{array}$$

The decomposition at a node function is

$$\begin{array}{l} \mathit{dcmp}(k, n) = \\ \quad \mathit{decompose}(\mathit{outpred}(k, n), \mathit{avoid}(k)) \end{array}$$

The derivable before a node predicate is implemented as

$$\begin{array}{l} \mathit{der}(k, n, t) = \\ \quad \textbf{let } T_p, T_a = \mathit{dcmp}(k, n) \textbf{ in} \\ \quad \mathit{bld}(t, T_p, T_a) \end{array}$$

Discussion

Add discussion.

10 Carried Only Within

A set of encryptions T_e protects critical message t in message t' if t is carried by t' only within a member of T_e. The definition of the carried only within (COW) relation to follow makes this concept precise. The concept is used when solving authentication tests (Section 11).

The question of the protection of a critical message is posed within the context of a given pruned skeleton k. The message algebra is $\mathfrak{A}_\top(X)$, where X is the finite variable set $\mathit{vars}(k)$. Thus all substitutions in this section are finite maps.

Definition 10.1 (Ancestors)**.** Let $t' = t \mathbin{@} p$. The *ancestors* of t' in t at p is the set $\mathit{anc}(t, p) = \{t \mathbin{@} p' \mid p' \text{ a proper prefix of } p\}$.

Definition 10.2 (Carried Only Within)**.** Message t is *carried only within* set T_e in t', written $t \odot^{T_e} t'$, if for all carried positions p of t in t', there exists an ancestor $t_a \in \mathit{anc}(t', p)$ and $t_e \in T_e$ such that $t_a \equiv t_e$. The negation, $\neg(t \odot^{T_e} t')$ is written $t \dagger^{T_e} t'$.

$cows(t, T, t') =$
  $cows_0(t, T, t', \sigma_{\text{id}})$ — σ_{id} is the identity subst

$cows_0(t, T, t', \sigma) =$
  if $\sigma(t)$ is COW $\sigma(T)$ at $\sigma(t')$ **then**
    $\{\sigma\}$
  else
    let $S = fold(t, T, t', \sigma)$ **in**
    $\bigcup_{\sigma' \in S} cows_0(t, T, t', \sigma')$

$fold(t, T, t', \sigma) =$
  $\{\sigma' \circ \sigma \mid \sigma' \in fold_0(\sigma(T), \sigma(t'), \{\sigma_{\text{id}}\}, carpos(\sigma(t), \sigma(t')))$

$fold_0(T, t', S, \{\}) = S$
$fold_0(T, t', S, \langle p \rangle \frown P) =$
  $fold_0(T, t', solve(anc(t', p), T, S), P)$

$solve(T, T', S) =$
  $\{\sigma' \mid t \in T, t' \in T', \sigma \in S, \sigma' \in unify(t, t', \sigma)\}$

Figure 6: The *cows* Function

Definition 2.11 defines $carpos(t, t')$, the set of positions at which t' carries t.

The details of a reduction on skeletons called a augmentation will be described in Section 12. In simplified form, for an augmentation, given t, T_e, and t', one must find all most general unifiers σ such that $\sigma(t)$ is carried only within set $\sigma(T_e)$ in $\sigma(t')$.

A carried only within solution cannot be directly computed using Definition 10.2. Given terms t_a and t_e, the *unify* function specified at the end of Section 2 finds substitutions σ such $\sigma(t_a) \equiv \sigma(t_e)$, however, the carried positions $carpos(\sigma(t), \sigma(t'))$, are used before the *unify* function computes the substitution σ. Figure 6 displays the iterative procedure that breaks the cyclic dependencies. Each step of the iteration improves an approximation of a solution to the problem. The correctness of this function is the subject a paper in preparation.

11 Authentication Tests

In what follows, we assume all skeletons are pruned, and use the word "skeleton" to mean pruned skeleton.

Definition 11.1 (Protectors)**.** Let *deriv* be a boolean valued function that determines if a message is derivable. The encryptions that protect t_c in t is $\mathit{protectors}(\mathit{deriv}, t_c, t) = \mathit{prot}(t)$ where

$$\mathit{prot}(t) = \begin{cases} \text{undefined} & \text{if } t \equiv t_c, \text{ else} \\ \{\} & \text{if } t = \{\!|t_0|\!\}_{t_1} \text{ and } t_c \text{ is not carried by } t_0, \text{ else} \\ \{\{\!|t_0|\!\}_{t_1}\} & \text{if } t = \{\!|t_0|\!\}_{t_1} \text{ and } \neg\, \mathit{deriv}(\mathit{inv}(t_1)), \text{ else} \\ \mathit{prot}(t_0) & \text{if } t = \{\!|t_0|\!\}_{t_1}, \text{ else} \\ \bigcup_{i<n} \mathit{prot}(t_i) & \text{if } t = f(t_0, \ldots, t_{n-1}) \text{ and } t \text{ is not an atom, else} \\ \{\} & \text{otherwise.} \end{cases}$$

Definition 11.2 (Escape Set)**.** The escape set for message t_c at n in skeleton k is the set of encryptions $\mathit{esc}(k, n, t_c)$ where

$$\mathit{esc}(k, n, t_c) = \{t_e \mid t_e \in \mathit{protectors}(\lambda t.\, \mathit{der}(k, n, t), t_c, t_o), t_o \in \mathit{outpred}(k, n)\}$$

and $\mathit{der}(k, n, t)$ is true when t is derivable before n in k (See Definition 9.3).

The *der* function is implemented as $\mathit{der}(k, n, t) = \mathit{bld}(t, T_p, T_a)$ where $(T_p, T_a) = \mathit{dcmp}(k, n)$, so that T_p and T_a need not be recomputed.

Definition 11.3 (Critical Position)**.** Position p is a *critical position* of $t = \mathit{msg}(k, n)$ if

1. p is a carried position in t,
2. either $t \mathbin{@} p \in \mathit{unique}(k)$ and $t \mathbin{@} p$ originates in k, or $t \mathbin{@} p = \{\!|t_0|\!\}_{t_1}$ and t_1 is not derivable before n in k,
3. $\mathit{esc}(k, n, t \mathbin{@} p)$ is defined, and
4. $\mathit{anc}(t, p) \cap \mathit{esc}(k, n, t \mathbin{@} p) = \emptyset$.

The message at a critical position is called a *critical message*. It is a *critical nonce* if the message is an atom, otherwise it is a *critical encryption*. Observe that every critical message at a node in a skeleton is not derivable at the node.

Theorem 11.1. A reception node is unrealized iff it has a critical position.

Definition 11.4 (Critical Position Solved)**.** Suppose p is a critical position at n_0 in k_0 and $k_0 \stackrel{\phi,\sigma}{\longmapsto} k_1$. Let $t_0 = msg(k, n) \mathbin{@} p$, $t_1 = \sigma(t_0)$, $T = \sigma(esc(k_0, n_0, t_0))$, $n_1 = \phi(n_0)$, and $t = msg(k_1, n_1)$. Critical position p is solved in k_1 after k_0 at n_0 if:

1. $anc(t, p) \cap T \neq \emptyset$, or
2. for some $t_p \in outpred(k_1, n_1)$, t_1 is not carried only within T in t_p, or
3. the decryption key of a member of T is derivable before n_1 in k_1, or
4. t_1 is an encryption and its encryption key is derivable before n_1 in k_1.

Definition 11.5 (Contraction)**.** Let p be a critical position at n in k, $t = msg(k, n)$, and $T_e = esc(k, n, t \mathbin{@} p)$. Suppose there is a substitution σ such that for some $t_a \in anc(t, p)$, $t_e \in T_e$, $\sigma(t_a) = \sigma(t_e)$. Skeleton k_1 is a *contraction* if $k \xrightarrow{\mathbb{S}_\sigma} k_0 \xrightarrow{skel} k_1$.

CPSA computes a set of substitutions for each critical position, and then removes some substitutions to form a complete set of most general unifiers. Only most general unifiers are used for contractions.

The function $\mathbb{A}_{i,n}$ augments a preskeleton with a new strand. It appends the instance i to the sequence of instances, adds node orderings, and adds atoms as specified by the role of the instance. The function orders the last node in the strand before some node n in the preskeleton.

Definition 11.6 (Augmentation)**.** Skeleton k_0 reduces to preskeleton k_1 by the augmentation i, n, written $k_0 \xrightarrow{\mathbb{A}_{i,n}} k_1$ if

1. $insts(k_1) = insts(k_0) \frown \langle i \rangle$;
2. $\prec_{k_1} = (\prec_{k_0} \cup \Rightarrow_{k_1} \cup \{(n_0, n)\})^*$, where $n_0 = (|\, insts(k_0)|, height(i) - 1))$;
3. $unique(k_1)$ is $unique(k_0)$ and the inherited uniquely originating atoms in i;
4. $non(k_1)$ is $non(k_0)$ and the inherited non-originating atoms in i.

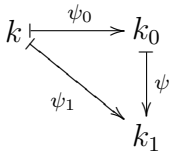
Definition 11.7 (Regular Augmentation)**.** Let t_c be a critical message at n in k, and i be an instance of a regular, non-listener role. Skeleton k_2 is a *regular augmentation* if $k \xrightarrow{\mathbb{S}_\sigma} k_0 \xrightarrow{\mathbb{A}_{i,n}} k_1 \xrightarrow{skel} k_2$ for some substitution σ and t_c is solved in k_2 after k_0 at n.

The details of regular augmentation is the subject of the next section.

Definition 11.8 (Listener Augmentation)**.** Let t_c be a critical message at n in k, and $T = esc(k, n, t_c)$. For each $\{|t_0|\}_{t_1} \in T$, skeleton k_1 is a *listener augmentation* if $k \xrightarrow{\mathbb{A}_{i,n}} k_0 \xrightarrow{skel} k_1$ and i is a listener for $inv(t_1)$. If $t_c = \{|t_0|\}_{t_1}$, then skeleton k_1 is a *listener augmentation* if $k \xrightarrow{\mathbb{A}_{i,n}} k_0 \xrightarrow{skel} k_1$ and i is a listener for t_1.

Definition 11.9 (Cohort Member)**.** For unrealized node n in a skeleton k_0, and a position p at n, $k_0 \xrightarrow{n,p} k_1$ asserts that k_1 is a member of the cohort of k_0, where k_1 is derived using contraction, regular augmentation, or listener augmentation, and p is solved in k_1 after k_0 at n. For the setwise cohort member reduction, $\{k_0\} \xrightarrow{n,t} \{k_1 \mid k_0 \xrightarrow{n,p} k_1\}$, when n is unrealized in k_0, and p is a critical position at n.

Theorem 11.2 (Cohort Correct)**.** Let node n be unrealized in skeleton k, and p be a critical position at n. For every realized skeleton k_1 such that $k \xmapsto{\psi_1} k_1$, there is a k_0, ψ_0, and ψ with $k \xrightarrow{n,p} k_0$, $k \xmapsto{\psi_0} k_0$, and $\psi_1 = \psi \circ \psi_0$.



Theorem 11.3 (Cohort)**.** Reduction $\xrightarrow{co}\!\!\!\!\rightarrow = \bigcup_{n,t} \xrightarrow{n,p}\!\!\!\!\rightarrow$ is confluent.

Theorem 11.4 (Critical Message Solved)**.** If $k_0 \xrightarrow{n_0,p_0} k_1 \xrightarrow{n_1,p_1} \ldots \xrightarrow{n_{\ell-1},p_{\ell-1}} k_\ell$ is a sequence of cohort member reductions, then for positive ℓ, p_0 is solved in k_ℓ after k_0 at n_0.

12 Finding Regular Augmentations

Let t_c be the critical message that demonstrates n is a test node in skeleton k. For each substitution-instance pair (σ, i) that satisfies some properties, there

$$
\begin{array}{l}
\mathit{cowt}(t, T, C, S) = \\
\quad \bigcup_{\sigma \in S} \mathit{cowt}_0(t, T, C, \sigma) \\
\\
\mathit{cowt}_0(t, T, C, \sigma) = \\
\quad \textbf{if } \forall t. \pm t \in C \rightarrow \sigma(t) \text{ is COW } \sigma(T) \text{ at } \sigma(t') \textbf{ then} \\
\qquad \{\sigma\} \\
\quad \textbf{else} \\
\qquad \mathit{cowt}(t, T, C, \mathit{foldn}(t, T, C, \{\sigma\})) \\
\\
\mathit{foldn}(t, T, \langle\rangle, S) = S \\
\mathit{foldn}(t, T, \langle \pm t' \rangle \frown C, S) = \\
\quad \mathit{foldn}(t, T, C, \bigcup_{\sigma \in S} \mathit{fold}(t, T, t', \sigma))
\end{array}
$$

Figure 7: The *cowt* Function

is a potential regular augmentation with $\xrightarrow{skel} \circ \xrightarrow{\mathbb{A}_{i,n}} \circ \xrightarrow{\mathbb{S}_\sigma}$. When successful, the message t in the last node of the added strand is outbound, carries $\sigma(t_c)$, but $\sigma(t_c)$ is not carried only within escape set $\sigma(T_e)$ in t. Moreover, for every other message t in the strand, $\sigma(t_c)$ is carried only within escape set $\sigma(T_e)$ in t. The last node in the strand is called a *transforming node*, as this node no longer protects the critical message, but nodes that precede it do.

When generating a candidate substitution-instance pair (σ, i) for augmentation, $\mathit{trace}(i)$ must contain a member of T_t, the set of *target messages*. The critical message t_c is a member along with each ancestor of the critical message t_c on every path to a place at which t_c is carried in every member of the escape set T_e, with the exception of the members of T_e. That is, $T_t = \{t_c\} \cup \{t_t \mid t_e \in T_e, p \in \mathit{carpos}(t_c, t_e), t_t \in \mathit{anc}(t_e, p)\} \setminus T_e$.

To find all candidate substitution-instance pairs, each role in the protocol is considered. For each role r, a substitution σ_r is created. The domain of σ_r, $\mathit{Dom}(\sigma_r)$, is the variables that occur in the role's trace. The substitution σ_r is a bijection, and each variable in $\mathit{Ran}(\sigma_r)$ is chosen in a way that ensures it does not occur in the skeleton k or in any of its roles.

Consider each outbound message t_{n-1} in $\mathit{rtrace}(r) = \langle \pm t_0, \ldots, +t_{n-1}, \ldots \rangle$. Let substitutions S be a complete set of most general unifiers σ' such that for every message t that is carried by t_{n-1} and message $t_t \in T_t$, $\sigma'(t) \equiv \sigma'(t_t)$, and $\sigma_r \trianglelefteq \sigma'$. This operation inserts the critical message into the trace $\sigma_r \circ \mathit{rtrace}(r)$.

The next step is to explore ways to extend the substitution σ' so that for

events in $\sigma' \circ rtrace(r)$ that precede t_{n-1}, the critical message is carried only within the escape set. The function *cowt*, presented in Figure 7, performs the explorations, producing the substitutions $S' = cowt(t_c, T_e, rtrace(r)|_{n-1}, S)$. Function *fold* is defined in Figure 6.

The final step is to remove substitutions $\sigma \in S'$ such that $\sigma(t_c)$ is carried only within $\sigma(T_e)$ in $\sigma(t_{n-1})$ and produce a complete set of most general unifiers S'' by removing less general unifiers. For each $\sigma' \in S''$, there is a potential regular augmentation constructed from (σ, i), where $role(i) = r$, $height(i) = n$, $subst(i) = \sigma$, and σ is σ' stripped of mappings to messages with variables that do not occur in $\sigma' \circ rtrace(r)|_n$.

For target terms to be the reasonable set for insertion of the critical message, one must require that variables of sort message are acquired. This fact needs to be explained and noted as another reason for the acquired variable constraint.

12.1 Regular Augmentation and Hulling

Consider regular augmentation $k \xrightarrow{\mathbb{S}_\sigma} k_0 \xrightarrow{\mathbb{A}_{i,n}} k_1 \xrightarrow{skel} k_2$, where k_1 is not a hulled preskeleton. Let k_3 be the result of one hulling step. It is crucial that $k \longmapsto k_2$, but it need not be the case that $k_1 \longmapsto k_3$. The reason is that a point of origination on the augmenting strand may move during hulling, but that's okay since the augmenting strand is not in the image of a homomorphism from k. An implementation must be careful not to check the preservation of origination points in this particular case.

13 Generalization

Each problem statement for CPSA is expressed as a preskeleton. If the preskeleton cannot be transformed into a single skeleton using the Preskeleton Reduction System, an error is signaled. Otherwise, the first skeleton is designated as the point-of-view skeleton. For each skeleton generated from a point-of-view skeleton via contraction, augmentation, and generalization, there is a homomorphism from the point-of-view skeleton. Simplifying the implementation is the motivation for restricting the algorithm to problem statements that are expressed by a single skeleton.

Definition 13.1 (Generalize)**.** A skeleton k_0 *generalizes* skeleton k_1, written

$k_1 \xrightarrow{<}_k k_0$, if both k_0 and k_1 are realized, k_0 and k_1 are not isomorphic, there is a homomorphism from a point-of-view skeleton k to k_0, and a strandwise injective homomorphism $k_0 \mapsto k_1$.

Recent experiments show that pruning must not be performed when generalizing.

If skeletons are allowed to be isomorphic, we write $k_1 \xrightarrow{\leq}_k k_0$, and note that $\xrightarrow{\leq}_k$ defines a partial ordering. Therefore, there are maximal elements in the partial ordering. A shape associated with a preskeleton is a maximally generalized realized skeleton derived from the preskeleton.

Definition 13.2 (Shape)**.** Let k_0 be a preskeleton such that $k_0 \xrightarrow{skel} k$ and k is unique, and let k_1 be a realized skeleton such that $k \mapsto k_1$. Skeleton k_2 is a *shape* of k_0 if $k_1 \xrightarrow{\leq}_k k_2$, and k_2 is maximal among skeletons that generalize k_1.

There are four generalization reductions used to transform a realized skeleton into its shapes: deletion, weakening, forgetting, and separation.

Definition 13.3 (Deletion)**.** Skeleton k_0 *generalizes by deletion* skeleton k_1, written $k_1 \xrightarrow{\mathbb{D}_n}_k k_0$, if $k_1 \xrightarrow{<}_k k_0$, $k_2 \xrightarrow{skel} k_0$, and k_2 is the result of deleting node n in k_1 and all of the nodes that follow it in its strand.

Definition 13.4 (Weakening)**.** Skeleton k_0 *generalizes by weakening* skeleton k_1, written $k_1 \xrightarrow{\mathbb{W}_{n,n'}}_k k_0$, if $k_1 \xrightarrow{<}_k k_0$, $k_2 \xrightarrow{skel} k_0$, and k_2 is k_1 except $\prec_{k_2} = (\prec_{k_1} \setminus \{(n, n')\})^*$.

Definition 13.5 (Forgetting)**.** Skeleton k_0 *generalizes by origination assumption forgetting* skeleton k_1, written $k_1 \xrightarrow{\mathbb{F}_t}_k k_0$, if $k_1 \xrightarrow{<}_k k_0$, $k_2 \xrightarrow{skel} k_0$, and k_2 is k_1 except $unique(k_2) = unique(k_1) \setminus \{t\}$ and $non(k_2) = non(k_1) \setminus \{t\}$.

Sometimes a more general skeleton can be found by replacing some occurrences of one variable by a fresh variable. For variable separation, the location of an occurrence of a variable is defined using a skeleton's instance sequence.

Definition 13.6 (Location)**.** Message t is at *location* (s, x, p) in k if $t = subst(I(s))(x) @ p$ and $I = insts(k)$.

Definition 13.7 (Separation)**.** Skeleton k_0 *generalizes by variable separation* skeleton k_1, written $k_1 \xrightarrow{\mathbb{V}_t}_k k_0$, if $k_1 \xrightarrow{<}_k k_0$, $k_2 \xrightarrow{skel} k_0$, and k_2 is k_1 except t is a variable that occurs in multiple locations in k_1, and k_2 is the result of replacing t with a variable t_0 of the same sort at a subset of t's locations, where t_0 occurs nowhere in k_1.

When separating a non-originating term, both the term and its clone are non-originating. When separating a uniquely originating term, either the term or its clone is uniquely originating.

What happens when separating t in k into t and t_0, and $\mathsf{ltk}(t,t) \in non(k)$? Should a skeleton k_0 with $\mathsf{ltk}(t,t_0) \in non(k_0)$ be a candidate separation? Currently, only skeletons k_1 with $\mathsf{ltk}(t,t) \in non(k_1)$ and $\mathsf{ltk}(t_0,t_0) \in non(k_1)$ are considered.

Definition 13.8 (Generalization)**.** The reduction $\xrightarrow{gen}_k = \bigcup_n \xrightarrow{\mathbb{D}_n}_k \cup \bigcup_{n,n'} \xrightarrow{\mathbb{W}_{n,n'}}_k \cup \bigcup_t \xrightarrow{\mathbb{F}_t}_k \cup \bigcup_t \xrightarrow{\mathbb{V}_t}_k$ is the *generalization* relation. For the setwise generalization reduction, $\{k_0\} \xrightarrow{gen}_k \{k_1\}$ when $k_0 \xrightarrow{gen}_k k_1$.

The fact that each generalization reduction replaces a singleton with just a singleton requires explanation. We're not sure it's justified, and harbor serious doubts. It is justified if we can prove the conjecture that the cohort reduction relation produces every shape, and generalization just identifies which realized skeletons are shapes. However, one member of the test suite provides a counter example to the conjecture. Barring a bug in CPSA*, the conjecture must be false.*

Theorem 13.1 (Generalization)**.** The relation $\xrightarrow{gen}_k$ is terminating.

Discussion

In [2], the shapes of a point-of-view skeleton are said to be minimal, in the partial ordering induced by injective homomorphism, among all realized homomorphic images of the point-of-view skeleton. Minimal corresponds to maximally generalized. The need for origination assumption forgetting was not known when [2] was written. Generalization by variable separation uses non-carried positions, and in particular, positions that traverse an atom edge. Algebras in previous strand space papers have no concept of a position that traverses an atom edge, and therefore cannot be used to specify generalization by variable separation.

14 Collapsing

Let reduction $\xrightarrow{cg}\!\!\!\!\!\rightarrow_k = (\xrightarrow{co}\!\!\!\!\!\rightarrow \cup \xrightarrow{gen}\!\!\!\!\!\rightarrow_k)^+$, the transitive closure of the cohort and generalization setwise reduction relations. The normal forms of this relation are sets of shapes, however, shapes may be missing from each set. The missing shapes are found by collapsing other shapes.

Definition 14.1 (Collapsing)**.** Let k_0 and k_1 be two skeletons such that there are two strands, s and s', and a substitution σ with $\sigma(evt(k_0, (s, p))) = \sigma(evt(k_0, (s', p)))$ for all $p < h$, where h is the height of strand s in k_0. Then k_0 *collapses* to k_1, written $k_0 \xrightarrow{clp} k_1$, if $k_0 \xrightarrow{\mathbb{S}_\sigma} k \xrightarrow{\mathbb{C}_{s,s'}} k' \xrightarrow{skel} k_1$.

Definition 14.2 (Setwise Collapsing)**.** For the collapsing relation $\xrightarrow{clp}\!\!\!\!\!\rightarrow_k \subseteq 2^{\mathfrak{K}} \times 2^{\mathfrak{K}}$, $K_0 \xrightarrow{clp}\!\!\!\!\!\rightarrow_k K_1$ if K_0 is a normal form of $\xrightarrow{cg}\!\!\!\!\!\rightarrow_k$, for some $k_0 \in K_0$, $k_0 \xrightarrow{clp} k$, $\{k\} \cup K_0 \xrightarrow{cg}\!\!\!\!\!\rightarrow_k K_1$, and $K_0 \neq K_1$.

Notice that a setwise cohort reduction may produce the empty set, but a setwise generalization and collapsing reduction never does.

15 Skeleton Reduction System

Let reduction $\twoheadrightarrow_k = \xrightarrow{cg}\!\!\!\!\!\rightarrow_k \cup (\xrightarrow{clp}\!\!\!\!\!\rightarrow_k)^+$.

Theorem 15.1. The reduction $\twoheadrightarrow_k$ is confluent.

Theorem 15.2 (Soundness)**.** Let k_0 be a preskeleton such that $k_0 \xrightarrow{skel} k$ and k is unique and unrealized. Skeleton k_1 is a shape of k_0 if $\{k\} \twoheadrightarrow_k K$, $k_1 \in K$, and K is a normal form.

Theorem 15.3 (Completeness)**.** Let k_0 be a preskeleton such that $k_0 \xrightarrow{skel} k$ and k is unique and unrealized. If $\{k\} \twoheadrightarrow_k K$, and K is a normal form, then k_1 is a shape of k_0 only if $k_1 \in K$.

A Programs Specified by a Role's Trace

The behavior associated with a role's trace is possible as long as some messages are available initially. The required initial messages are specified by a

pair of ternary relations, $T_0, C \rhd T_1$ and $T_0, \pm t \rhd T_1$. The relation $T_0, C \rhd T_1$ asserts that messages in T_1 are available after a run of C given the messages in T_0 are available initially. The relation is defined using the $T_0, \pm t \rhd T_1$ relation.

$$T, \langle\rangle \rhd T \qquad \frac{T_0, \pm t \rhd T \quad T, C \rhd T_1}{T_0, \langle \pm t \rangle \frown C \rhd T_1}$$

The $T_0, \pm t \rhd T_1$ relation is defined using the $T_0, C \rhd T_1$ relation. An outbound message can be formed if it is available initially

$$\frac{t \in T}{T, +t \rhd T}$$

or if it can be formed by construction.

$$\frac{T, \langle +t_0, \ldots, +t_{n-1} \rangle \rhd T}{T, +f(t_0, \ldots, t_{n-1}) \rhd T} \quad \left[\begin{array}{l} f \in \Sigma_{w,s} \wedge n = |w| \wedge \\ f(t_0, \ldots, t_{n-1}) \text{ not an atom} \end{array} \right]$$

An inbound message makes atoms, acquired variables, and encryptions available.

$$T, -t \rhd T \cup \{t\} \quad [t \text{ an atom or a variable}]$$

When the decryption key is available, the contents of the encryption is also available. Furthermore, the encryption can be sent in future messages without access to its encryption key.

$$\frac{T_0, + inv(t_1) \rhd T_0 \quad T_0, -t_0 \rhd T_1}{T_0, -\{\!|t_0|\!\}_{t_1} \rhd T_1 \cup \{\{\!|t_0|\!\}_{t_1}\}}$$

A received encryption that can be sent ensures the encyption agrees with currently available terms and makes nothing new available.

$$\frac{T, +\{\!|t_0|\!\}_{t_1} \rhd T}{T, -\{\!|t_0|\!\}_{t_1} \rhd T}$$

Consider an operation $f \in \Sigma_{w,s}$ other than the encryption operation. The order in which messages that occur in a message constructed using f are made available may determine if the decryption key of an encryption is available. Let π_n be a permutation on the domain of a sequence of length n.

$$\frac{T_0, \langle -t_0, \ldots, -t_{n-1} \rangle \circ \pi_n \rhd T_1}{T_0, -f(t_0, \ldots, t_{n-1}) \rhd T_1} \quad \left[\begin{array}{l} f \in \Sigma_{w,s} \wedge n = |w| \wedge f \neq \mathsf{enc} \wedge \\ f(t_0, \ldots, t_{n-1}) \text{ not an atom} \end{array} \right]$$

Definition A.1 (Trace Parameters)**.** The set of atoms T_0 are *parameters* of trace C if $T_0, C \rhd T_1$ for some T_1, and T_0 is minimal, that is for all T_0' such that $T_0', C \rhd T_1$, $T_0' \not\subset T_0$.

The role $\langle +\{\!|a, n|\!\}_{K_b}, -\{\!|n|\!\}_{K_a} \rangle$ has two sets of parameters, $\{a, n, K_b, K_a^{-1}\}$ and $\{a, n, K_b, K_a\}$. A program that implements the role using the second set is useless, so it will be ignored. The intended program for the role follows.

$\mathsf{proc}(a, n, K_b, K_a^{-1})$
  $\mathsf{send}(\{\!|a, n|\!\}_{K_b})$;
  $x_0 \leftarrow \mathsf{recv}()$;
  $x_1 \leftarrow \mathsf{decrypt}(x_0, K_a^{-1})$; — may fail
  $x_1 \neq n \rightarrow \mathsf{fail}$;
end

Let $T_0 = \{a, n, K_b, K_a^{-1}\}$ and $T_1 = T_0 \cup \{\{\!|n|\!\}_{K_a}\}$. The derivation tree that specifies the intended program has the following outline.

$\ldots$ construct outbound message
$T_0, +\{\!|a, n|\!\}_{K_b} \rhd T_0$
$\ldots$ decode inbound message
$T_0, -\{\!|n|\!\}_{K_a} \rhd T_1$
$\ldots$ compose trace
$T_0, \langle +\{\!|a, n|\!\}_{K_b}, -\{\!|n|\!\}_{K_a} \rangle \rhd T_1$

The decoding of the inbound message is specified by this derivation.

$$\frac{T_0, +K_a^{-1} \rhd T_0 \qquad \dfrac{T_0, +n \rhd T_0}{T_0, -n \rhd T_0}}{T_0, -\{\!|n|\!\}_{K_a} \rhd T_1}$$

Acknowledgment

Carolyn Talcott and Leonard Monk provided valuable feedback on drafts of this document.

References

[1] Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[2] 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 `http://eprint.iacr.org/2006/435`.

[3] Joseph A. Goguen and Jose Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105(2):217–273, 1992.

[4] Joshua D. Guttman and F. Javier Thayer. Authentication tests and the structure of bundles. *Theor. Comput. Sci.*, 283(2):333–380, 2002.

[5] Alan Robinson and Andrei Voronkov. *Handbook of Automated Reasoning*. The MIT Press, 2001.

[6] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1), 1999.

Index

Contents