Implementing Strand Space Algebras

John D. Ramsdell

The MITRE Corporation

October 2009

MITRE

Strand Spaces

Message: element of initial algebra $\mathfrak{A}_\Sigma$ over signature Σ

Event: transmission $(+t)$ or reception $(-t)$, $t \in \mathfrak{A}_\Sigma$

Trace: non-empty sequence of events $(\pm\mathfrak{A}_\Sigma)^+$
(a sequence of length n is a map from $\mathbb{Z}_n$)

Strand Space: strand set Θ and trace mapping $tr\colon \Theta \to (\pm\mathfrak{A}_\Sigma)^+$

Nodes

Nodes of (Θ, tr)**:** $\{(s,p) \mid s \in \Theta, 0 \leq p < |\, tr(s)|\}$

Event at node: $evt(s,p) = tr(s)(p)$

Strand succession: $\{(s,p-1) \Rightarrow (s,p) \mid s \in \Theta, 0 < p < |\, tr(s)|\}$

Bundles

Bundle $\mathcal{B}(\Theta, tr, \rightarrow)$**:** a directed acyclic graph such that

Vertices: nodes of (Θ, tr) [a finite set]

Edges: communication ($\rightarrow$) and strand succession ($\Rightarrow$)

- $n_0 \rightarrow n_1$ implies $evt(n_0) = +t$ and $evt(n_1) = -t$
- for each reception node n_1, there is a unique transmission node n_0 with $n_0 \rightarrow n_1$

Roles and Protocols

Role variables: variable set X

Role message: element of free algebra $\mathfrak{A}_\Sigma(X)$

- generated by variable set X

Role: non-empty sequence of events $(\pm\mathfrak{A}_\Sigma(X))^+$

Protocol: set of roles with pairwise disjoint variable sets

Runs of Protocols

A bundle $\mathcal{B}(\Theta, tr, \rightarrow)$ is a run of a protocol P if there is a role mapping $rl\colon \Theta \rightarrow P$ such that:

- $s \in \Theta$ implies $|\,tr(s)| \leq |\,rl(s)|$

- $s \in \Theta, 0 \leq p < tr(s)$ implies $tr(s)(p) = \sigma(rl(s)(p))$
  for some Σ-homomorphism $\sigma\colon \mathfrak{A}_\Sigma(X) \rightarrow \mathfrak{A}_\Sigma$,
  where X is the union of the protocol's role variable sets

Free Algebras

Let $\mathcal{K}$ be a class of Σ-algebras. The Σ-algebra $\mathcal{A}$ is called *free in $\mathcal{K}$ with generating set X* iff

- $\mathcal{A}$ is generated by $X \subseteq \mathcal{A}$,
- $\mathcal{A} \in \mathcal{K}$, and
- for every Σ-algebra $\mathcal{B}$ in $\mathcal{K}$, every mapping $\sigma\colon X \to \mathcal{B}$ can be extended to a homomorphism $\hat{\sigma}\colon \mathcal{A} \to \mathcal{B}$.

— Baader and Nipkow, "Term Rewriting and All That", 1998

Strand Spaces Over Free Algebras

- Protocols mention few constants used in a run
- So use Σ' which is Σ without the constants ignored by roles
- Consider strand spaces and bundles over free algebra $\mathfrak{A}_{\Sigma'}(Y)$

  Algebra $\mathfrak{A}_{\Sigma}$

  - *satisfies* $\mathcal{B}(\Theta, tr, \rightarrow)$ over algebra $\mathfrak{A}_{\Sigma'}(Y)$
  - *with* Σ-homomorphism $\sigma\colon \mathfrak{A}_{\Sigma}(Y) \rightarrow \mathfrak{A}_{\Sigma}$
  - because $\mathcal{B}(\Theta, \sigma \circ tr, \rightarrow)$ is a bundle over algebra $\mathfrak{A}_{\Sigma}$

Interpreting Received Messages

Pair (M, N)**:** receiver may extract both M and N

Encryption $\{\!|M|\!\}_N$**:** receiver may extract M
if decryption key possessed

Inverse asymmetric key M^{-1}**:** receiver may never extract M

The implementation uses sorts and sort sensitive predicates on terms to correctly interpret received messages.

Basic Crypto Signature

Sort symbols: mesg, name, text, data, skey, akey

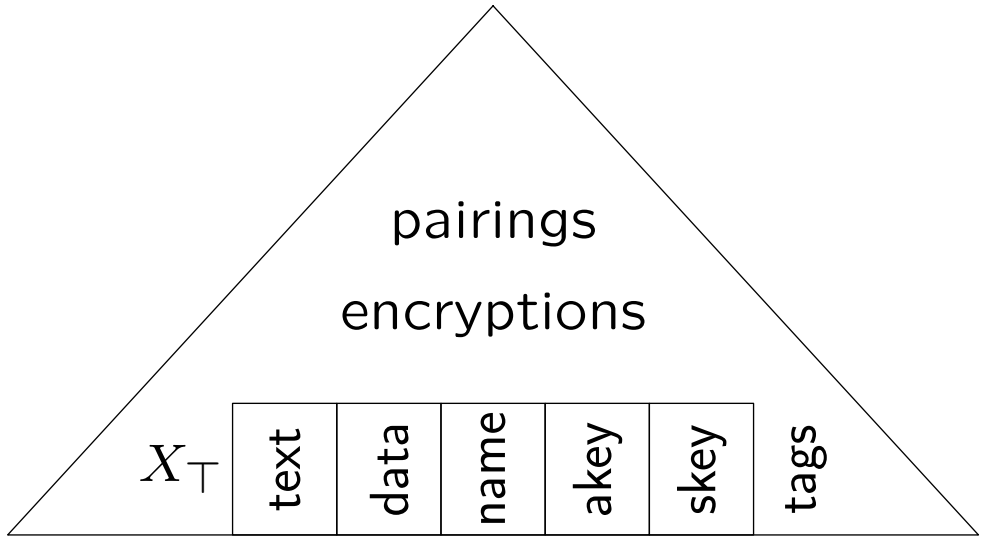
Subsorts: name, text, data, akey, skey $<$ mesg

$\{\!\lvert \cdot \rvert\!\}_{(\cdot)}$: mesg $\times$ mesg $\rightarrow$ mesg	Encryption
$(\cdot, \cdot)$: mesg $\times$ mesg $\rightarrow$ mesg	Pairing
"..." : mesg	Tag constants
$K_{(\cdot)}$: name $\rightarrow$ akey	Public key of name
$(\cdot)^{-1}$: akey $\rightarrow$ akey	Inverse of asymmetric key
ltk: name $\times$ name $\rightarrow$ skey	Long term shared key

Identity: $(x^{-1})^{-1} = x$ for x: akey

For pairing, parentheses are omitted when the context permits, and comma is right associative. Sort $\top$ is an alias for mesg.

Carrier Sets of Basic Crypto Algebra $\mathfrak{A}_\Sigma(X)$



Carrier set of sort mesg

$X_\top$: variables of sort mesg

Base sort: a subsort of mesg, e.g. akey

Atom: an element of the carrier set of a base sort, e.g. M^{-1}

Inverse Key

The decryption key of encryption $\{|t_0|\}_{t_1}$ is $inv(t_1)$, where

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

Interpreting Role Messages

- Roles abstract programs, strands abstract their runs
- Consider a role with reception event $-\{|M|\}_N$
- What program is specified by this event in a role?
  - Use decryption key associated with N to extract M, or
  - Test if $\{|M|\}_N$ is equal to a known message

Interpretions of a Role

- Given an initial set of messages, what programs are associated with a trace?

- Use ternary relations $T_0, C \triangleright T_1$ and $T_0, \pm t \triangleright T_1$, where $T_0, C \triangleright T_1$ asserts $t \in T_1$ is available after a run of C starting with messages T_0 and

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

  where $\frown$ denotes sequence concatenation

Interpretions of a Transmission

- A message can be sent if it is known

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

- A message can be sent if it can be constructed

$$\frac{T, \langle +t_0, \ldots, +t_{n-1} \rangle \rhd T}{T, +f(t_0, \ldots, t_{n-1}) \rhd T} \quad [f(t_0, \ldots, t_{n-1}) \text{ not an atom}]$$

Interpretions of a Reception

- Receive an atom or variable and make it known

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

- Receive an encryption by decrypting it

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

- Receive an encryption ensuring it matches a known message

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

- Receive a message by decomposition

$$\frac{T_0, \langle -t_0, \ldots, -t_{n-1} \rangle \circ \pi_n \rhd T_1}{T_0, -f(t_0, \ldots, t_{n-1}) \rhd T_1} \left[\begin{array}{l} \pi_n \text{ is a permutation} \\ f(t_0, \ldots, t_{n-1}) \text{ not an atom} \end{array} \right]$$

Role 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 $T_0', C \rhd T_1$, $T_0' \not\subset T_0$
- Program `cpsaparameters` computes the parameter sets of roles
- Found flaw in CAVES attestation protocol in draft submitted for publication

Hashing and Parameter Sets

- Hashing simulated by encryption
  - Decryption key available to no one
  - Asymmetric key h^{-1} assumed non-originating in roles
- Hashing macro: $\#M \Longrightarrow \{|\text{“hash”}, M|\}_h$
  - Tag “hash” ensures hashes are not confused with other encryptions
- Key h^{-1} must not be in every parameter set of some role
  - Was in erroneous version of CAVES

Finding Contractions

- The core operation in CPSA is solving authentication tests
  - A test node has a reception event $-t$ the penetrator cannot construct
  - Penetrator has access to a set of encryptions T_e, but not their decryption keys
  - To solve the test, CPSA infers additional regular behavior
- One way to solve a test is to find a contraction, meaning
  - Find a substitution σ such that given $\sigma(T_e)$, the penetrator can construct $\sigma(t)$
  - Contraction algorithm is tricky—presented next

Positions

- A *position* p is a finite sequence of natural numbers
- The term in t that *occurs at* p, written $t @ p$, is:

$$t @ \langle\rangle = t;$$
$$f(t_0, \ldots, t_{n-1}) @ \langle i \rangle \frown p = t_i @ p \text{ if } i < n$$

- Position p *traverses a key edge* in t if $t @ p_0 = \{\!|t_0|\!\}_{t_1}$ and $p = p_0 \frown \langle 1 \rangle \frown p_1$
- Position p *traverses an atom edge* in t if $t @ p_0 = f(t_0, \ldots, t_{n-1})$ is an atom and $p = p_0 \frown \langle i \rangle \frown p_1$, where $i < n$
- A position is a *carried position* in t if it traverses no key or atom edge in t

Carried Positions

Given a term t, the set of positions at which t' carries t is $carpos(t, t')$, where

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

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

Carried Only Within

- When $t' = t \mathbin{@} p$, the *ancestors* of t' in t at p are

$$anc(t, p) = \{t \mathbin{@} p' \mid p' \text{ a proper prefix of } p\}$$

- Term t is *carried only within* set T_e in t', if

$$\forall p \in carpos(t, t'). \exists t_a t_e. t_a \in anc(t', p) \wedge t_e \in T_e \wedge t_a \equiv t_e$$

  Note that $|carpos(t, t')|$ equations must be satisfied

- For every test, reception message t' carries a term t such that t is not carried only within T_e in t'

Carried Only Within Solutions

The set S of substitutions is a *carried only within solution* for t, T_e, and t' iff $\sigma \in S$ implies $\sigma(t)$ is carried only within $\sigma(T_e)$ in $\sigma(t')$ and S is a complete set of most general unifiers.

Why are solutions tricky to compute? A solution requires

- solving a system of equations of cardinality $|\mathit{carpos}(\sigma(t), \sigma(t'))|$
- but σ cannot be computed without knowing the equations

Compute by finding a fixed point of a function

Carried Only Within at a Substitution

- A key property of $carpos$

  Lemma 1. $carpos(t, t') \subseteq carpos(\sigma(t), \sigma(t'))$

- Term t is *carried only within* T_e in t' *at* σ if

  $$\forall p \in carpos(t, t').\, \exists t_a t_e.\, t_a \in anc(t', p) \wedge t_e \in T_e \wedge \sigma(t_a) \equiv \sigma(t_e)$$

Solving Systems of Equations

- Extending a solution by solving one additional equation

  $$unify\colon \mathfrak{A}_\top(X) \times \mathfrak{A}_\top(X) \times (X \to \mathfrak{A}_\top(X)) \to \mathsf{Set}(X \to \mathfrak{A}_\top(X))$$
  $$unify(t, t', \sigma) = \{\sigma' \circ \sigma \mid \sigma' \in unify(\sigma(t), \sigma(t'), \sigma_{\mathsf{id}})\}$$

  where σ_{id} is the identity substitution

- Extending solutions by solving one extra equation from the cross product

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

Carried Only Within at a Substitution Implementation

Find substitutions that solve an equation for each carried position

$$
\begin{aligned}
&fold(t, T_e, t', S) = \\
&\quad fold_0(t, T_e, t', S, carpos(t, t'))
\end{aligned}
$$

$$
\begin{aligned}
&fold_0(t, T_e, t', S, \{\}) = S \\
&fold_0(t, T_e, t', S, \{p\} \cup P) = \\
&\quad fold_0(t, T_e, t', solve(anc(t', p), T_e, S), P \setminus \{p\})
\end{aligned}
$$

Term t is carried only within T_e in t' at σ iff $\sigma \in fold(t, T_e, t', \{\sigma_{\mathsf{id}}\})$

Carried Only Within Solutions

Implementation

Iterate $fold$ to find a fixed point

$cows(t, T_e, t') =$
  $cows_0(t, T_e, t', \sigma_{\mathsf{id}})$ — σ_{id} is the identity substitution

$cows_0(t, T_e, t', \sigma) =$
  if t is carried only within T_e at t' **then**
    $\{\sigma\}$
  else
    let $S = fold(t, T_e, t', \{\sigma\})$ **in**
    $\bigcup_{\sigma' \in S} cows_0(\sigma'(t), \sigma'(T_e), \sigma'(t'), \sigma')$

Correctness

Soundness: $\sigma \in cows(t, T_e, t')$ implies $\sigma(t)$ is carried only within $\sigma(T_e)$ in $\sigma(t')$

Termination: $cows(t, T_e, t')$ terminates on all inputs

Completeness: $\sigma(t)$ is carried only within $\sigma(T_e)$ in $\sigma(t')$ implies there exists a substitution σ' such that $\sigma' \trianglelefteq \sigma$ and $\sigma' \in cows(t, T_e, t')$

Substitution σ_0 is *more general than* σ_1, written $\sigma_0 \trianglelefteq \sigma_1$, if there exists a substitution σ_2 such that $\sigma_1 = \sigma_2 \circ \sigma_0$

Diffie-Hellman Signature

New base sorts:

base, expn $<$ mesg

New operations:

g: base	Generator
$(\cdot)^{(\cdot)}$: base $\times$ expn $\longrightarrow$ base	Exponentiation
$(\cdot\cdot)$: expn $\times$ expn $\longrightarrow$ expn	Multiplication
1: expn	Unit
$1/(\cdot)$: expn $\longrightarrow$ expn	Reciprocal

New equations:

$(h^x)^y \approx h^{xy}$	
$h^1 \approx h$	
$xy \approx yx$	Commutativity
$x(yz) \approx (xy)z$	Associativity
$1x \approx x$	Identity
$x/x \approx 1$	Cancellation