Musings on Protocols and Monads

ArmoredSoftware Team
palexand@ku.edu

May 29, 2015

Abstract

This document captures discussions on formally representing protocols using monadic constructs. This is a living document and will be updated frequently.

Notation

Throughout we use a trivial monadic notation for protocols that serves as our "assembly language" target for protocol compilation. The following conventions hold:

```
do {                % evaluate functions in sequence
     f(x);          % calculate f(x) and discard the result
     y <- f(x);     % calculate f(x) and bind the result to y
     send a $ x;    % evaluate x and send the result to a
     y <- receive a % receive data from a and the result to y
}
```

This is early work, so we play fast and loose with specific syntax and semantics. `send` and `receive` operate synchronously. Each `send` must have a corresponding `receive` to complete its operation.

Example Protocols

Needham-Schroeder-Lowe

Message Sequence

$$A \rightarrow B : \{N_A, A\}_{B^+}$$
$$B \rightarrow A : \{N_A, N_B, B\}_{A^+}$$
$$A \rightarrow B : \{N_B\}_{B^+}$$

Monadic Representation

```
Notes:
-Identifiers "a" and "b" serve as principal identity handles
-na, nb, m1, m2, m3 are variables
-Assume Na and Nb are generated fresh by principals A and B respectively for each run
-Only A can decrypt using a, and B likewise with b
-Public keys are known a priori


Principal A:
do {
     m1 <- encrypt({Na, a}, b);     % encrypt a's nonce and its i.d. with b's public key
     send b $ m1;                   % send result to b
     m2 <- receive b;               % receive (encrypted) message from b
     (na, nb, x) <- decrypt(m2, a); % decrypt m2 using private key of a
     m3 <- encrypt(nb, b);          % encrypt b's nonce with b's public key
     send b $ m3;                   % send result to b
}

Principal B:
do {
     m1 <- receive                  % receive initial (encrypted) message
     (na, a) <- decrypt(m1, b);     % decrypt m1 using b's private key
     m2 <- encrypt({na, Nb, b}, a); % build m2 using na and a
     send a $ m2;                   % send result to a
     m3 <- receive a;               % receive (encrypted) message from b
     (nb) <- decrypt(m3, b);        % decrypt it
}
```


Wide-Mouthed Frog

Message Sequence

$$A \rightarrow S : \{N_A, B, K_{AB}\}_{K_{AS}}$$
$$S \rightarrow B : \{N_S, A, K_{AB}\}_{K_{BS}}$$

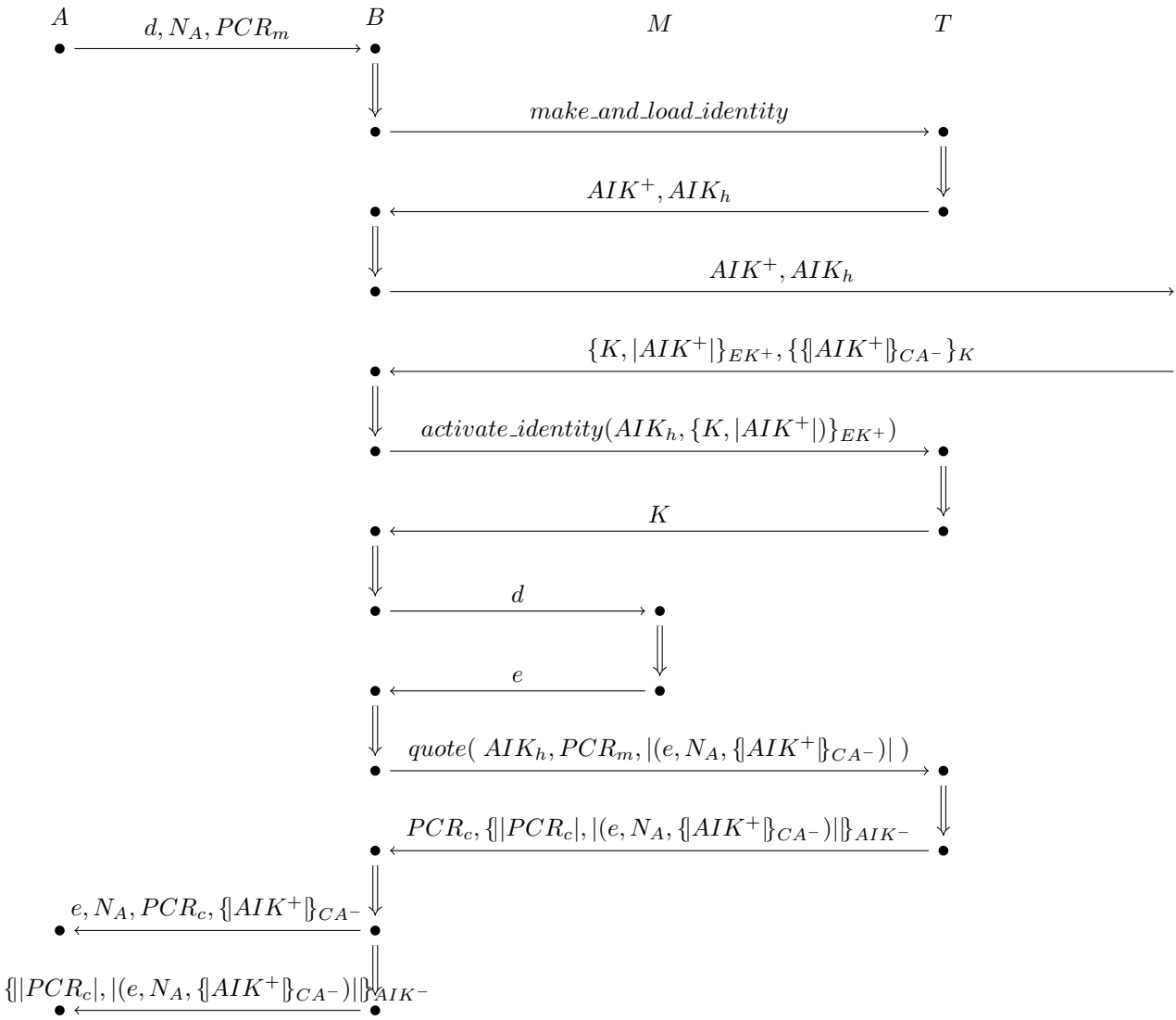
Monadic Representation

CA Protocol

Message Sequence

$$
\begin{aligned}
&A \to B : d, N_A, PCR_m \\
&B \to T : make\_and\_load\_identity \\
&T \to B : AIK_h \\
&B \to C : B, AIK^+ \\
&C \to B : \{K, |AIK|\}_{EK^+}, \{\{|AIK^+|\}_{CA^-}\}_{K^+} \\
&B \to T : activate\_identity(AIK_h, |AIK|) \\
&T \to B : K \\
&B \to M : d \\
&M \to B : e \\
&B \to T : quote(\ AIK_h, PCR_m, |(e, N_A, \{|AIK^+|\}_{CA^-})|\ ) \\
&T \to B : PCR_c, \{||PCR_c|, |(e, N_A, \{|AIK^+|\}_{CA^-})|\}_{AIK^-} \\
&B \to A : e, N_A, PCR_c, \{|AIK^+|\}_{CA^-} \\
&B \to A : \{||PCR_c|, |(e, N_A, \{|AIK^+|\}_{CA^-})|\}_{AIK^-}
\end{aligned}
$$

Strand Space Diagram



Monadic Representation