ArmoredSoftware Semantics 0.0

ArmoredSoftware Crew
Information and Telecommunication Technology Center
The University of Kansas
palexand@ku.edu

December 5, 2014

Contents

List of Figures

Abstract

This document describes evolving ARMOREDSOFTWARE semantic definitions.

1 Introduction

2 SPI Calculus

Examples motivated by **?**.

2.1 Wide Mouth Frog

2.2 Needham Schroeder

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

$$A \triangleq \overline{c}\langle\{(A, N_A)\}_{B+}\rangle.$$
$$c(M).$$
$$\text{case } \{M\}_{A-} \text{ of } (N_A, nb) \text{ in}$$
$$\overline{c}\langle\{nb\}_{B+}\rangle.$$
$$A$$
$$B \triangleq c(M).$$
$$\text{case } \{M\}_{B-} \text{ of } (x, n) \text{ in}$$
$$\overline{c}\langle\{(n, N_B)\}_{x+}\rangle.$$
$$c(M).$$
$$\text{case } \{M\}_{B-} \text{ of } N_B \text{ in } B$$
$$sys \triangleq (\nu c)A \mid B$$

React Inter $\dfrac{}{\overline{m}\langle N\rangle.P \mid m(x).Q \to P \mid [x \to N]Q}$

Red Replace $\dfrac{}{!P > P \mid !P}$

Red Match $\dfrac{}{[\![M \text{ is } M]\!]P > P}$

Red Let $\dfrac{}{\text{let } (x, y) = (M, N) \text{ in } P > [x \to M][y \to N]P}$

Note that we may want a more general let that matches more than pairs here. We'll see what the other inference rules give us.

Red Zero $\dfrac{}{\text{case } 0 \text{ of } 0 : P \; suc(x) : Q > P}$

Red Suc $\dfrac{}{\text{case } suc(M) \text{ of } 0 : P \; suc(x) : Q > [x \to M]Q}$

I find the case rules over naturals quite crude.

Red Sym Decrypt $\dfrac{}{\text{case } \{M\}_k \text{ of } \{x\}_k \text{ in } P > [x \to M]P}$

Additional proposed semantic rules for public/private key encryption and signature checking

Red Asym Decrypt 1 $\dfrac{}{\text{case } \{M\}_{k+} \text{ of } \{x\}_{k-} \text{ in } P > [x \to M]P}$

Red Asym Decrypt 2 $\dfrac{}{\text{case } \{M\}_{k-} \text{ of } \{x\}_{k+} \text{ in } P > [x \to M]P}$

The previous two rules capture the essence of asymmetric key pairs. Specifically, encrypt with one and decrypt with the other.

Assume $|M|$ is the hash and not the message itself and $\{|M|\}_{k-}$ is the hash encrypted with private key, k^-. Thus, a signed message is the pair $(M, \{|M|\}_{k-})$

consisting of the message and the signed hash. Given this, a successful signature check looks something like this:

$$\text{let } (m,s) = (M, \{|M|\}_{k-}) \text{ in case } s \text{ of } \{x\}_{k+} \text{ in } [\![x \text{ is } |m|]\!]P$$
$$> [m \to M][s \to \{|M|\}_{k-}]\text{case } s \text{ of } \{x\}_{k+} \text{ in } [\![x \text{ is } |m|]\!]P$$
$$> \text{case } \{|M|\}_{k-} \text{ of } \{x\}_{k+} \text{ in } [m \to M][s \to \{|M|\}_{k-}][\![x \text{ is } |m|]\!]P$$
$$> [x \to |M|][m \to M][s \to \{|M|\}_{k-}][\![x \text{ is } |m|]\!]P$$
$$> [\![|M| \text{ is } |M|]\!][x \to |M|][m \to M][s \to \{|M|\}_{k-}]P$$
$$> [x \to |M|][m \to M][s \to \{|M|\}_{k-}]P$$

This is precisely what we want. Specifically, P with m replaced by the message M and s replaced by the decrypted signature, $|M|$, produced by the signature check. It is unlikely that $|M|$ will be used in P, but it is available.

If the signature does not match, the process hangs. Assume the hash is incorrect. Specifically, $M \neq N$:

$$\text{let } (m,s) = (M, \{|N|\}_{k-}) \text{ in case } s \text{ of } \{x\}_{k+} \text{ in } [\![x \text{ is } |m|]\!]P$$
$$> [m \to M][s \to \{|N|\}_{k-}]\text{case } s \text{ of } \{x\}_{k+} \text{ in } [\![x \text{ is } |m|]\!]P$$
$$> \text{case } \{|N|\}_{k-} \text{ of } \{x\}_{k+} \text{ in } [m \to M][s \to \{|N|\}_{k-}][\![x \text{ is } |m|]\!]P$$
$$> [x \to |N|][m \to M][s \to \{|M|\}_{k-}][\![x \text{ is } |m|]\!]P$$
$$> [\![|N| \text{ is } |M|]\!][x \to |N|][m \to M][s \to \{|M|\}_{k-}]P$$

The process is stuck when $|N|$ is $|M|$ fails because $N \neq M$.

Now assume the wrong private key was used to sign the message hash. Specifically, $j \neq k$:

$$\text{let } (m,s) = (M, \{|M|\}_{j-}) \text{ in case } s \text{ of } \{x\}_{k+} \text{ in } [\![x \text{ is } |m|]\!]P$$
$$> [m \to M][s \to \{|M|\}_{j-}]\text{case } s \text{ of } \{x\}_{k+} \text{ in } [\![x \text{ is } |m|]\!]P$$
$$> \text{case } \{|M|\}_{j-} \text{ of } \{x\}_{k+} \text{ in } [m \to M][s \to \{|M|\}_{k-}][\![x \text{ is } |m|]\!]P$$

The process is stuck when $\{|M|\}_{j-}$ does not unify with $\{x\}_{k+}$.

Do we really want a signature check that fails to get stuck? I think so. M is available, but the signature check is stuck. A signed message is best represented as a pair $(M, \{|M|\}_{k-})$ allowing the message to be explicitly available.

$$\text{Struct Nil}\frac{}{P \mid \mathbf{0} \equiv P}$$

$$\text{Struct Comm} \frac{}{P \mid Q \equiv Q \mid P}$$

$$\text{Struct Assoc} \frac{}{P \mid (Q \mid R) \equiv (P \mid Q) \mid R}$$

$$\text{Struct Switch} \frac{}{(\nu m)(\nu n)P \equiv (\nu n)(\nu m)P}$$

$$\text{Struct Drop} \frac{}{(\nu n)\mathbf{0} \equiv \mathbf{0}}$$

$$\text{Struct Extrusion} \frac{n \notin fv(P)}{(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q}$$

$$\text{Struct Red} \frac{P > Q}{P \equiv Q}$$

$$\text{Struct Refl} \frac{}{P \equiv P}$$

$$\text{Struct Symm} \frac{P \equiv Q}{Q \equiv P}$$

$$\text{Struct Trans} \frac{P \equiv Q \qquad Q \equiv R}{P \equiv R}$$

$$\text{Struct Par} \frac{P \equiv P'}{P \mid Q \equiv P' \mid Q}$$

$$\text{Struct Res} \frac{P \equiv P'}{(\nu n)P \equiv (\nu n)P'}$$

$$\text{React Struct} \frac{P \equiv P' \qquad P' \to Q' \qquad Q' \equiv Q}{P \to Q}$$

$$\text{React Par} \frac{P' \to P'}{P \mid Q \to P' \mid Q}$$

$$\text{React Res} \frac{P' \to P'}{(\nu n)P \to (\nu n)P}$$

$$A \quad \triangleq \quad \overline{c}\langle \{(A, N\_A)\}_{B^+}\rangle.$$
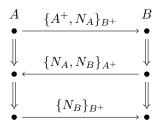
2.3   Privacy CA Protocol

3   Strand Spaces

3.1   Needham Schroder

$$A \to B : \{A^+, N_A\}_{B^+} \text{ on } c$$
$$B \to A : \{N_A, N_B\}_{A^+} \text{ on } c$$
$$A \to B : \{N_B\}_{B^+} \text{ on } c$$

$$A \quad \xrightarrow{\{A^+, N_A\}_{B+}} \quad B$$

$$\xleftarrow{\{N_A, N_B\}_{A+}}$$

$$\xrightarrow{\{N_B\}_{B+}}$$

A Glossary

- **0** - null process
- $|M|$ - hash of M
- K^+ - public half of asymmetric key K
- K^- - private half of asymmetric key K
- $\{M\}_K$ - encrypt M with symmetric key K
- $\{M\}_{K+}$ - encrypt M with the public key from K
- $\{M\}_{K-}$ - decrypt M with the public key from K
- $\{|M|\}_{K-}$ - sign M with the private key from K
- $\{|M|\}_{K+}$ - check signature on M with the public key from K
- $(\nu x)P$ - new variable x defined in scope of P
- $\bar{c}\langle M \rangle$ - send M on channel c
- $c(M)$ - receive M on channel c
- $!P$ - infinite replication of P
- $P + Q$ - P or Q
- $P \mid Q$ - P in parallel with Q
- case $\{M\}_k$ of x in P - attempt to decrypt $\{M\}_k$ and bind to x in P if successful. Stuck if unsuccessful
- case $\{M\}_{k-}$ of x in P - attempt to decrypt $\{M\}_{k+}$ and bind to x in P if successful. Stuck if unsuccessful
- case $\{|M|\}_{k+}$ of x in P - attempt to check signature $\{|M|\}_{k-}$ and bind to x in P if successful. Stuck if unsuccessful
- case x of y $0 : P$ $suc(x) : Q$ - case splitting over integers. x is bound in Q.
- let $(x, y) = M$ in y - match M to (x, y) binding x and y to pair elements in M
- $A \stackrel{\triangle}{=} B$ - define an equivalence
- $A \rightarrow B : M$ on c - A sends B message M on channel c

$$A \stackrel{\triangle}{=} (\nu c) \, \bar{c}\langle M \rangle.\mathbf{0} \mid$$
$$c(M).A$$