Homework 13

Chirag Sachdev

Week 13

Abstract

This project is a part of HW12 of Assurance Foundations. The homework deals with integration of ML and HOL to LaTeX. The goal of this report is to show reproducibility which is the groundwork for credibility that I have done this on my own without any external help. Every Chapter demonstrates the following sections:

- Problem Statement
- Relevant Code
- Test Results

This project includes the following packages:

634format.sty A format style for this course

listings Package for displaying and inputting ML source code

holtex HOL style files and commands to display in the report

This document also demonstrates my ability to :

- Easily generate a table of contents,
- Refer to chapter and section labels

My skills and my professional details can be found at `https://www.linkedin.in/in/chiragsachdev`.

Acknowledgments

I would gratefully acknowledge Dr. Shiu-Kai Chin and my other professors at Syracuse University and my Professors at Drexel University for being the wonderful mentors they are to guide me through my journey of obtaining a Master's Dregree.

Contents

Chapter 1

Executive Summary

All requirements for this project are satisfied. Specifically we prove the following theorems:

[SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma]

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG inputOKr2 SM0StateInterp
       (certsr2a npriv privcmd (PR privcmd))
       (mapSM0r1input
          (mapSM0inputOperatorBob
             (Name (Role Commander) says
              prop (SOME (PR privcmd))))::ins) s outs) ⇒
  (M,Oi,Os) sat prop NONE
```

[SM0r2_Commander_Alice_trap_privcmd_justified_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOKr2 SM0StateInterp
         (certsr2a npriv privcmd (PR privcmd))
         (Name (KeyS (pubK Alice)) quoting
          Name (Role Commander) says prop (SOME (PR privcmd))::
              ins) s outs)
      (CFG inputOKr2 SM0StateInterp
         (certsr2a npriv privcmd (PR privcmd)) ins
         (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⟺
    inputOKr2
      (Name (KeyS (pubK Alice)) quoting
       Name (Role Commander) says prop (SOME (PR privcmd))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOKr2 SM0StateInterp
         (certsr2a npriv privcmd (PR privcmd))
         (Name (KeyS (pubK Alice)) quoting
          Name (Role Commander) says prop (SOME (PR privcmd))::
              ins) s outs) ∧ (M,Oi,Os) sat prop NONE
```

[SM0r2_Commander_mapSM0r1input_trap_privcmd_justifed_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOKr2 SM0StateInterp
         (certsr2a npriv privcmd (PR privcmd))
         (mapSM0r1input
            (mapSM0inputOperatorBob
               (Name (Role Commander) says
                prop (SOME (PR privcmd))))::ins) s outs)
      (CFG inputOKr2 SM0StateInterp
         (certsr2a npriv privcmd (PR privcmd)) ins
         (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⟺
```

```
      inputOKr2
        (mapSMOr1input
           (mapSMOinputOperatorBob
              (Name (Role Commander) says
               prop (SOME (PR privcmd))))) ∧
      CFGInterpret (M,Oi,Os)
        (CFG inputOKr2 SMOStateInterp
           (certsr2a npriv privcmd (PR privcmd))
           (mapSMOr1input
              (mapSMOinputOperatorBob
                 (Name (Role Commander) says
                  prop (SOME (PR privcmd))))::ins) s outs) ∧
      (M,Oi,Os) sat prop NONE
```

[certificatesr3a_certsr2a_map_thm]

```
 ⊢ ∀npriv privcmd.
      MAP certificateInterpret
        (certificatesr3a npriv privcmd (PR privcmd)) =
      certsr2a npriv privcmd (PR privcmd)
```

[SMOr3_mkinMsg_SMOr2_Alice_Commander_trap_privcmd_lemma]

```
 ⊢ CFG2Interpret (M,Oi,Os)
      (CFG2 MsgInterpret certificateInterpret inputOKr2
         (certificatesr3a npriv privcmd (PR privcmd))
         SMOStateInterp
         (mkinMsg
            (mapSMOr1input
               (mapSMOinputOperatorBob
                  (Name (Role Commander) says
                   prop (SOME (PR privcmd)))))::ins) s outs) ⇒
    (M,Oi,Os) sat prop NONE
```

[SMOr3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm]

```
 ⊢ ∀NS Out M Oi Os.
      TR2 (M,Oi,Os) (trap (PR privcmd))
        (CFG2 MsgInterpret certificateInterpret inputOKr2
           (certificatesr3a npriv privcmd (PR privcmd))
           SMOStateInterp
           (mkinMsg
              (mapSMOr1input
                 (mapSMOinputOperatorBob
                    (Name (Role Commander) says
                     prop (SOME (PR privcmd)))))::ins) s outs)
        (CFG2 MsgInterpret certificateInterpret inputOKr2
           (certificatesr3a npriv privcmd (PR privcmd))
           SMOStateInterp ins (NS s (trap (PR privcmd)))
           (Out s (trap (PR privcmd))::outs)) ⟺
      inputOKr2
        (MsgInterpret
           (mkinMsg
              (mapSMOr1input
                 (mapSMOinputOperatorBob
                    (Name (Role Commander) says
                     prop (SOME (PR privcmd)))))))) ∧
      CFG2Interpret (M,Oi,Os)
        (CFG2 MsgInterpret certificateInterpret inputOKr2
           (certificatesr3a npriv privcmd (PR privcmd))
```

```
        SMOStateInterp
        (mkinMsg
           (mapSMOr1input
              (mapSMOinputOperatorBob
                 (Name (Role Commander) says
                  prop (SOME (PR privcmd)))))::ins) s outs) ∧
    (M,Oi,Os) sat prop NONE
```

[SMOr3_Commander_Alice_privcmd_trap_privcmd_justified_thm]

```
⊢ ∀NS Out M Oi Os.
    TR2 (M,Oi,Os) (trap (PR privcmd))
      (CFG2 MsgInterpret certificateInterpret inputOKr2
         (certificatesr3a npriv privcmd (PR privcmd))
         SMOStateInterp
         (MSG Alice (Order Commander (PR privcmd))
            (sign (privK Alice)
               (hash (SOME (Order Commander (PR privcmd)))))::
              ins) s outs)
      (CFG2 MsgInterpret certificateInterpret inputOKr2
         (certificatesr3a npriv privcmd (PR privcmd))
         SMOStateInterp ins (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⟺
    inputOKr2
      (MsgInterpret
         (MSG Alice (Order Commander (PR privcmd))
            (sign (privK Alice)
               (hash
                  (SOME (Order Commander (PR privcmd)))))))) ∧
    CFG2Interpret (M,Oi,Os)
      (CFG2 MsgInterpret certificateInterpret inputOKr2
         (certificatesr3a npriv privcmd (PR privcmd))
         SMOStateInterp
         (MSG Alice (Order Commander (PR privcmd))
            (sign (privK Alice)
               (hash (SOME (Order Commander (PR privcmd)))))::
              ins) s outs) ∧ (M,Oi,Os) sat prop NONE
```

[SMOr3_Alice_TR2_iff_TR_trap_privcmd]

```
⊢ ∀NS Out M Oi Os.
    TR2 (M,Oi,Os) (trap (PR privcmd))
      (CFG2 MsgInterpret certificateInterpret inputOKr2
         (certificatesr3a npriv privcmd (PR privcmd))
         SMOStateInterp
         (mkinMsg
            (mapSMOr1input
               (mapSMOinputOperatorBob
                  (Name (Role Commander) says
                   prop (SOME (PR privcmd)))))::ins2) s outs)
      (CFG2 MsgInterpret certificateInterpret inputOKr2
         (certificatesr3a npriv privcmd (PR privcmd))
         SMOStateInterp ins2 (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⟺
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOKr2 SMOStateInterp
         (certsr2a npriv privcmd (PR privcmd))
         (Name (KeyS (pubK Alice)) quoting
          Name (Role Commander) says prop (SOME (PR privcmd))::
              ins) s outs)
```

```
(CFG inputOKr2 SMOStateInterp
   (certsr2a npriv privcmd (PR privcmd)) ins
   (NS s (trap (PR privcmd)))
   (Out s (trap (PR privcmd))::outs))
```

[Reproducibility in ML and LaTeX]
The ML and LaTeX source files compile with no errors.

Chapter 2

Problem statement

Prove the following theorems in SM0r2:

[SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma]

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG inputOKr2 SM0StateInterp
       (certsr2a npriv privcmd (PR privcmd))
       (mapSM0r1input
          (mapSM0inputOperatorBob
             (Name (Role Commander) says
              prop (SOME (PR privcmd))))::ins) s outs) ⇒
  (M,Oi,Os) sat prop NONE
```

[SM0r2_Commander_Alice_trap_privcmd_justified_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOKr2 SM0StateInterp
         (certsr2a npriv privcmd (PR privcmd))
         (Name (KeyS (pubK Alice)) quoting
          Name (Role Commander) says prop (SOME (PR privcmd))::
              ins) s outs)
      (CFG inputOKr2 SM0StateInterp
         (certsr2a npriv privcmd (PR privcmd)) ins
         (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⟺
    inputOKr2
      (Name (KeyS (pubK Alice)) quoting
       Name (Role Commander) says prop (SOME (PR privcmd))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOKr2 SM0StateInterp
         (certsr2a npriv privcmd (PR privcmd))
         (Name (KeyS (pubK Alice)) quoting
          Name (Role Commander) says prop (SOME (PR privcmd))::
              ins) s outs) ∧ (M,Oi,Os) sat prop NONE
```

[SM0r2_Commander_mapSM0r1input_trap_privcmd_justifed_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOKr2 SM0StateInterp
         (certsr2a npriv privcmd (PR privcmd))
         (mapSM0r1input
            (mapSM0inputOperatorBob
               (Name (Role Commander) says
                prop (SOME (PR privcmd))))::ins) s outs)
      (CFG inputOKr2 SM0StateInterp
         (certsr2a npriv privcmd (PR privcmd)) ins
         (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⟺
```

```
      inputOKr2
        (mapSM0r1input
           (mapSM0inputOperatorBob
              (Name (Role Commander) says
               prop (SOME (PR privcmd))))) ∧
      CFGInterpret (M, Oi, Os)
        (CFG inputOKr2 SM0StateInterp
           (certsr2a npriv privcmd (PR privcmd))
           (mapSM0r1input
              (mapSM0inputOperatorBob
                 (Name (Role Commander) says
                  prop (SOME (PR privcmd))))::ins) s outs) ∧
      (M, Oi, Os) sat prop NONE
```

Prove the following theorems in SM0r3:

[certificatesr3a_certsr2a_map_thm]

```
⊢ ∀ npriv privcmd.
    MAP certificateInterpret
      (certificatesr3a npriv privcmd (PR privcmd)) =
    certsr2a npriv privcmd (PR privcmd)
```

[SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma]

```
⊢ CFG2Interpret (M, Oi, Os)
    (CFG2 MsgInterpret certificateInterpret inputOKr2
       (certificatesr3a npriv privcmd (PR privcmd))
       SM0StateInterp
       (mkinMsg
          (mapSM0r1input
             (mapSM0inputOperatorBob
                (Name (Role Commander) says
                 prop (SOME (PR privcmd)))))::ins) s outs) ⇒
  (M, Oi, Os) sat prop NONE
```

[SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm]

```
⊢ ∀ NS Out M Oi Os.
    TR2 (M, Oi, Os) (trap (PR privcmd))
      (CFG2 MsgInterpret certificateInterpret inputOKr2
         (certificatesr3a npriv privcmd (PR privcmd))
         SM0StateInterp
         (mkinMsg
            (mapSM0r1input
               (mapSM0inputOperatorBob
                  (Name (Role Commander) says
                   prop (SOME (PR privcmd)))))::ins) s outs)
      (CFG2 MsgInterpret certificateInterpret inputOKr2
         (certificatesr3a npriv privcmd (PR privcmd))
         SM0StateInterp ins (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⟺
    inputOKr2
      (MsgInterpret
         (mkinMsg
            (mapSM0r1input
               (mapSM0inputOperatorBob
                  (Name (Role Commander) says
                   prop (SOME (PR privcmd))))))) ∧
    CFG2Interpret (M, Oi, Os)
      (CFG2 MsgInterpret certificateInterpret inputOKr2
```

```
          (certificatesr3a npriv privcmd (PR privcmd))
          SMOStateInterp
          (mkinMsg
             (mapSMOr1input
                (mapSMOinputOperatorBob
                   (Name (Role Commander) says
                    prop (SOME (PR privcmd)))))::ins) s outs) ∧
     (M,Oi,Os) sat prop NONE
```

[SMOr3_Commander_Alice_privcmd_trap_privcmd_justified_thm]

```
 ⊢ ∀NS Out M Oi Os.
     TR2 (M,Oi,Os) (trap (PR privcmd))
       (CFG2 MsgInterpret certificateInterpret inputOKr2
          (certificatesr3a npriv privcmd (PR privcmd))
          SMOStateInterp
          (MSG Alice (Order Commander (PR privcmd))
             (sign (privK Alice)
                (hash (SOME (Order Commander (PR privcmd))))::
              ins) s outs)
       (CFG2 MsgInterpret certificateInterpret inputOKr2
          (certificatesr3a npriv privcmd (PR privcmd))
          SMOStateInterp ins (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) ⟺
     inputOKr2
       (MsgInterpret
          (MSG Alice (Order Commander (PR privcmd))
             (sign (privK Alice)
                (hash
                   (SOME (Order Commander (PR privcmd)))))))) ∧
     CFG2Interpret (M,Oi,Os)
       (CFG2 MsgInterpret certificateInterpret inputOKr2
          (certificatesr3a npriv privcmd (PR privcmd))
          SMOStateInterp
          (MSG Alice (Order Commander (PR privcmd))
             (sign (privK Alice)
                (hash (SOME (Order Commander (PR privcmd))))::
              ins) s outs) ∧ (M,Oi,Os) sat prop NONE
```

[SMOr3_Alice_TR2_iff_TR_trap_privcmd]

```
 ⊢ ∀NS Out M Oi Os.
     TR2 (M,Oi,Os) (trap (PR privcmd))
       (CFG2 MsgInterpret certificateInterpret inputOKr2
          (certificatesr3a npriv privcmd (PR privcmd))
          SMOStateInterp
          (mkinMsg
             (mapSMOr1input
                (mapSMOinputOperatorBob
                   (Name (Role Commander) says
                    prop (SOME (PR privcmd)))))::ins2) s outs)
       (CFG2 MsgInterpret certificateInterpret inputOKr2
          (certificatesr3a npriv privcmd (PR privcmd))
          SMOStateInterp ins2 (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) ⟺
     TR (M,Oi,Os) (trap (PR privcmd))
       (CFG inputOKr2 SMOStateInterp
          (certsr2a npriv privcmd (PR privcmd))
          (Name (KeyS (pubK Alice)) quoting
           Name (Role Commander) says prop (SOME (PR privcmd))::
```

```
        ins) s outs)
(CFG inputOKr2 SMOStateInterp
   (certsr2a npriv privcmd (PR privcmd)) ins
   (NS s (trap (PR privcmd)))
   (Out s (trap (PR privcmd))::outs))
```

Chapter 3

SM0r2 Solutions

3.1 Proof of SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_-lemma

3.1.1 Relevant Code

```
val th1 =
TAC_PROOF(([],
``((M,Oi,Os) satList (certsr2a npriv privcmd (PR privcmd))) ==>
  (((M:(command inst,'b, principal,'d,'e)Kripke),Oi,Os)
  sat Name (KeyS (pubK Alice)) speaks_for Name (Staff Alice))``),
REWRITE_TAC[certsr2a_def,certsr2root_def,certsr2signed_def,
            certsr1a_def,certs_def,satList_CONS,
            (GSYM satList_conj),satList_nil] THEN
PROVE_TAC[Derived_Speaks_For,Controls])

val th2 =
TAC_PROOF(
([],``(((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
        sat (Name (Role Commander)) speaks_for (Name (Role Commander)))``),
PROVE_TAC[Idemp_Speaks_For])

val th3 =
TAC_PROOF(
([],``(((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
       satList (certsr2a npriv privcmd (PR privcmd))) ==>
      ((M,Oi,Os) sat (((Name (KeyS (pubK Alice))) quoting
                       (Name (Role Commander)))
                      speaks_for
                     ((Name (Staff Alice))
                     quoting (Name (Role Commander)))))``),
PROVE_TAC[Mono_speaks_for,th1,Derived_Speaks_For,th2])

val th4 =
TAC_PROOF(
([],``(((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
       satList (certsr2a npriv privcmd(PR privcmd))) ==>
((M,Oi,Os) sat ((((Name (KeyS (pubK Alice)))
quoting (Name (Role Commander))) says
       (prop (SOME (PR privcmd)))):
       (command inst,principal,'d,'e)Form)) ==>
      ((M,Oi,Os) sat ((((Name (Staff Alice))
      quoting (Name (Role Commander))) says
       (prop (SOME (PR privcmd)))):
       (command inst,principal,'d,'e)Form))``),
PROVE_TAC[th3,Derived_Speaks_For])
```

```
val th5 =
TAC_PROOF(
([],``(((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
       satList (certsr2a npriv privcmd (PR privcmd))) ==>
((M,Oi,Os) sat (Name (Role Commander) says prop
(SOME (PR privcmd))) impf (prop NONE)) /\
      ((M,Oi,Os) sat (reps (Name (Staff Alice)) (Name (Role Commander))
       (prop (SOME (PR privcmd)))))``),
REWRITE_TAC[certsr2a_def,certsr2root_def,certsr2signed_def,certsr1a_def,
            certs_def,satList_CONS,satList_nil,(GSYM satList_conj)] THEN
PROVE_TAC[])


val th6 =
TAC_PROOF(([],
``(((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os) satList
   (certsr2a npriv privcmd (PR privcmd))) ==>
    ((M,Oi,Os) sat ((((Name (KeyS (pubK Alice)))
    quoting (Name (Role Commander))) says
       (prop (SOME (PR privcmd))))):(command inst,principal,'d,'e)Form))
  ==> ((M,Oi,Os) sat (prop NONE))``),
PROVE_TAC[Rep_Says,th4,th5,Modus_Ponens])

val SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma =
TAC_PROOF(([],
``CFGInterpret ((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
  (CFG inputOKr2 SM0StateInterp (certsr2a npriv privcmd (PR privcmd))
   (mapSM0r1input
    (mapSM0inputOperatorBob
     ((Name (Role Commander)) says (prop (SOME (PR (privcmd:privcmd))))))::ins)
   s (outs:output list)) ==>
  ((M,Oi,Os) sat (prop NONE))``),
REWRITE_TAC[CFGInterpret_def,mapSM0inputOperatorBob_def,mapSM0r1input_def] THEN
PROVE_TAC[th6])

val _ = save_thm("SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma",
SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma);
```

3.1.2 Session Transcript

1

```
> Meson search level: ......
Meson search level: ..
Meson search level: .....
Meson search level: .....
Meson search level: ....
Meson search level: ..........
Meson search level: ....
val SMOr2_mapSMOr1_Alice_Commander_trap_privcmd_lemma =
   |- CFGInterpret
     ((M :(command inst, 'b, principal, 'd, 'e) Kripke),(Oi :'d po),
      (Os :'e po))
     (CFG (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
        (SMOStateInterp :state ->
                         (command inst, principal, 'd, 'e) Form)
        (certsr2a (npriv :npriv) (privcmd :privcmd) (PR privcmd) :
           (command inst, principal, 'd, 'e) Form list)
        (mapSMOr1input
           (mapSMOinputOperatorBob
              (Name (Role Commander) says
               (prop (SOME (PR privcmd) :command inst) :
                  (command inst, principal, 'd, 'e) Form))::
             (ins :(command inst, principal, 'd, 'e) Form list))
        (s :state) (outs :output list)) ==>
   (M,Oi,Os) sat
   (prop (NONE :command inst) :(command inst, principal, 'd, 'e) Form):
   thm
val th1 =
   |- ((M :(command inst, 'b, principal, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) satList
   (certsr2a (npriv :npriv) (privcmd :privcmd) (PR privcmd) :
      (command inst, principal, 'd, 'e) Form list) ==>
   (M,Oi,Os) sat
   ((Name (KeyS (pubK Alice)) speaks_for Name (Staff Alice))
      :(command inst, principal, 'd, 'e) Form):
   thm
val th2 =
   |- ((M :(command inst, 'b, principal, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) sat
   ((Name (Role Commander) speaks_for Name (Role Commander))
      :(command inst, principal, 'd, 'e) Form):
   thm
val th3 =
   |- ((M :(command inst, 'b, principal, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) satList
   (certsr2a (npriv :npriv) (privcmd :privcmd) (PR privcmd) :
      (command inst, principal, 'd, 'e) Form list) ==>
   (M,Oi,Os) sat
   ((Name (KeyS (pubK Alice)) quoting Name (Role Commander) speaks_for
     Name (Staff Alice) quoting Name (Role Commander))
      :(command inst, principal, 'd, 'e) Form):
   thm
val th4 =
   |- ((M :(command inst, 'b, principal, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) satList
   (certsr2a (npriv :npriv) (privcmd :privcmd) (PR privcmd) :
      (command inst, principal, 'd, 'e) Form list) ==>
   (M,Oi,Os) sat
   Name (KeyS (pubK Alice)) quoting Name (Role Commander) says
   (prop (SOME (PR privcmd) :command inst) :
      (command inst, principal, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Staff Alice) quoting Name (Role Commander) says
   (prop (SOME (PR privcmd) :command inst) :
      (command inst, principal, 'd, 'e) Form):
   thm
val th5 =
   |- ((M :(command inst, 'b, principal, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) satList
   (certsr2a (npriv :npriv) (privcmd :privcmd) (PR privcmd) :
      (command inst, principal, 'd, 'e) Form list) ==>
   (M,Oi,Os) sat
   Name (Role Commander) says
   (prop (SOME (PR privcmd) :command inst) :
      (command inst, principal, 'd, 'e) Form) impf
   (prop (NONE :command inst) :
      (command inst, principal, 'd, 'e) Form) /\
   (M,Oi,Os) sat
   reps (Name (Staff Alice)) (Name (Role Commander))
     (prop (SOME (PR privcmd) :command inst) :
        (command inst, principal, 'd, 'e) Form):
   thm
val th6 =
   |- ((M :(command inst, 'b, principal, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) satList
   (certsr2a (npriv :npriv) (privcmd :privcmd) (PR privcmd) :
```

3.2 Proof of SM0r2_Commander_Alice_trap_privcmd_justified_thm

3.2.1 Relevant Code

```
val SM0r2_Commander_Alice_trap_privcmd_justified_thm =
let
 val th1 =
  ISPECL
 [``inputOKr2:(command inst, principal,'d,'e)Form -> bool``,
 ``SM0StateInterp:state->(command inst, principal,'d,'e)Form``,
``(certsr2a npriv privcmd (PR privcmd)):
  (command inst, principal,'d,'e)Form list``,
``(Name (KeyS (pubK Alice))) quoting (Name (Role Commander))``,``PR privcmd``,
           ``ins:(command inst,principal,'d,'e)Form list``,
             ``s:state``,``outs:output list``]
 TR_trap_cmd_rule
 val th2 =
 REWRITE_RULE[mapSM0inputOperatorBob_def,mapSM0r1input_def]
 SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma
in
   REWRITE_RULE[th2]th1
end

val _ = save_thm("SM0r2_Commander_Alice_trap_privcmd_justified_thm",
         SM0r2_Commander_Alice_trap_privcmd_justified_thm)
```

3.2.2 Session Transcript

```
> val SM0r2_Commander_Alice_trap_privcmd_justified_thm =                                    2
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, principal, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po).
     TR (M,Oi,Os) (trap (PR (privcmd :privcmd)))
       (CFG (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (SMOStateInterp :state ->
                           (command inst, principal, 'd, 'e) Form)
          (certsr2a (npriv :npriv) privcmd (PR privcmd) :
             (command inst, principal, 'd, 'e) Form list)
          (Name (KeyS (pubK Alice)) quoting Name (Role Commander) says
           (prop (SOME (PR privcmd) :command inst) :
              (command inst, principal, 'd, 'e) Form)::
               (ins :(command inst, principal, 'd, 'e) Form list))
          (s :state) (outs :output list))
       (CFG (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (SMOStateInterp :state ->
                           (command inst, principal, 'd, 'e) Form)
          (certsr2a npriv privcmd (PR privcmd) :
             (command inst, principal, 'd, 'e) Form list) ins
          (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) <=>
     inputOKr2
       (Name (KeyS (pubK Alice)) quoting Name (Role Commander) says
        (prop (SOME (PR privcmd) :command inst) :
           (command inst, principal, 'd, 'e) Form)) /\
     CFGInterpret (M,Oi,Os)
       (CFG (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (SMOStateInterp :state ->
                           (command inst, principal, 'd, 'e) Form)
          (certsr2a npriv privcmd (PR privcmd) :
             (command inst, principal, 'd, 'e) Form list)
          (Name (KeyS (pubK Alice)) quoting Name (Role Commander) says
           (prop (SOME (PR privcmd) :command inst) :
              (command inst, principal, 'd, 'e) Form)::ins) s outs) /\
     (M,Oi,Os) sat
     (prop (NONE :command inst) :(command inst, principal, 'd, 'e) Form):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***
```

3.3 Proof of SM0r2_Commander_mapSM0r1input_trap_privcmd_-justifed_thm

3.3.1 Relevant Code

```
val SM0r2_Commander_mapSM0r1input_trap_privcmd_justifed_thm =
let
 val th1 =
  TAC_PROOF(([],
   ``((Name (KeyS (pubK Alice))) quoting (Name (Role Commander))
      says (prop (SOME (PR (privcmd:privcmd))))
           :(command inst, principal,'d,'e)Form)
      =
      mapSM0r1input
         (mapSM0inputOperatorBob
         ((Name (Role Commander)) says (prop (SOME (PR (privcmd:privcmd)))))
                  :(command inst, principal,'d,'e)Form)``),
  PROVE_TAC[mapSM0r1input_def,mapSM0inputOperatorBob_def])
in
 REWRITE_RULE[th1]SM0r2_Commander_Alice_trap_privcmd_justified_thm
```

```
end

val _ = save_thm("SM0r2_Commander_mapSM0r1input_trap_privcmd_justifed_thm",
        SM0r2_Commander_mapSM0r1input_trap_privcmd_justifed_thm)
```

3.3.2 Session Transcript

3

```
> Meson search level: .......
val SM0r2_Commander_mapSM0r1input_trap_privcmd_justifed_thm =
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, principal, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po).
     TR (M,Oi,Os) (trap (PR (privcmd :privcmd)))
       (CFG (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (SM0StateInterp :state ->
                          (command inst, principal, 'd, 'e) Form)
          (certsr2a (npriv :npriv) privcmd (PR privcmd) :
             (command inst, principal, 'd, 'e) Form list)
          (mapSM0r1input
             (mapSM0inputOperatorBob
                (Name (Role Commander) says
                 (prop (SOME (PR privcmd) :command inst) :
                    (command inst, principal, 'd, 'e) Form)))::
               (ins :(command inst, principal, 'd, 'e) Form list))
          (s :state) (outs :output list))
       (CFG (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (SM0StateInterp :state ->
                          (command inst, principal, 'd, 'e) Form)
          (certsr2a npriv privcmd (PR privcmd) :
             (command inst, principal, 'd, 'e) Form list) ins
          (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) <=>
     inputOKr2
       (mapSM0r1input
          (mapSM0inputOperatorBob
             (Name (Role Commander) says
              (prop (SOME (PR privcmd) :command inst) :
                 (command inst, principal, 'd, 'e) Form)))) /\
     CFGInterpret (M,Oi,Os)
       (CFG (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (SM0StateInterp :state ->
                          (command inst, principal, 'd, 'e) Form)
          (certsr2a npriv privcmd (PR privcmd) :
             (command inst, principal, 'd, 'e) Form list)
          (mapSM0r1input
             (mapSM0inputOperatorBob
                (Name (Role Commander) says
                 (prop (SOME (PR privcmd) :command inst) :
                    (command inst, principal, 'd, 'e) Form)))::ins) s
          outs) /\
     (M,Oi,Os) sat
     (prop (NONE :command inst) :(command inst, principal, 'd, 'e) Form):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***

>
```

Chapter 4

SM0r3 Solutions

4.1 Proof of certificatesr3a_certsr2a_map_thm

4.1.1 Relevant Code

```
val certificatesr3a_certsr2a_map_thm =
TAC_PROOF(([],
``!(npriv :npriv) (privcmd :privcmd).
    MAP
      (certificateInterpret :('a, 'b) certificate ->
                             (command inst, principal, 'a, 'b) Form)
      (certificatesr3a npriv privcmd (PR privcmd) :
         ('a, 'b) certificate list) =
    (certsr2a npriv privcmd (PR privcmd) :
       (command inst, principal, 'a, 'b) Form list)``),
REWRITE_TAC[certificatesr3a_def,MAP_APPEND] THEN
REWRITE_TAC[MAP_certificateInterpret_mkSCert_thm,
 MAP_certificateInterpret_mkRCert_thm,certsr2a_def])

val _ = save_thm ("certificatesr3a_certsr2a_map_thm",
    certificatesr3a_certsr2a_map_thm)
```

4.1.2 Session Transcript

4

```
> val certificatesr3a_certsr2a_map_thm =
   |- !(npriv :npriv) (privcmd :privcmd).
     MAP
       (certificateInterpret :('a, 'b) certificate ->
                              (command inst, principal, 'a, 'b) Form)
       (certificatesr3a npriv privcmd (PR privcmd) :
          ('a, 'b) certificate list) =
     (certsr2a npriv privcmd (PR privcmd) :
        (command inst, principal, 'a, 'b) Form list):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***

>
```

4.2 Proof of SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma

4.2.1 Relevant Code

```
val SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma =
```

```
TAC_PROOF(([],
``CFG2Interpret
    ((M :(command inst, 'b, principal, 'd, 'e) Kripke),(Oi :'d po),
     (Os :'e po))
    (CFG2
       (MsgInterpret :inMsg -> (command inst, principal, 'd, 'e) Form)
       (certificateInterpret :('d, 'e) certificate ->
                              (command inst, principal, 'd, 'e) Form)
       (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
       (certificatesr3a (npriv :npriv) (privcmd :privcmd) (PR privcmd) :
          ('d, 'e) certificate list)
       (SM0StateInterp :state -> (command inst, principal, 'd, 'e) Form)
       (mkinMsg
          (mapSM0r1input
             (mapSM0inputOperatorBob
                (Name (Role Commander) says
                 (prop (SOME (PR privcmd) :command inst) :
                    (command inst, principal, 'd, 'e) Form))))::
            (ins :inMsg list)) (s :state) (outs :output list)) ==>
  (M,Oi,Os) sat
  (prop (NONE :command inst) :(command inst, principal, 'd, 'e) Form)``),
  REWRITE_TAC
 [CFG2Interpret_def,mapSM0inputOperatorBob_def,mapSM0r1input_def,
  MsgInterpret_inverts_mkinMsg_thm,
  certificatesr3a_certsr2a_map_thm] THEN
PROVE_TAC[th6])


val _ = save_thm("SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma",
        SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma)
```

4.2.2 Session Transcript

5

```
> Meson search level: ....
val SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma =
   |- CFG2Interpret
     ((M :(command inst, 'b, principal, 'd, 'e) Kripke),(Oi :'d po),
      (Os :'e po))
     (CFG2
        (MsgInterpret :inMsg -> (command inst, principal, 'd, 'e) Form)
        (certificateInterpret :('d, 'e) certificate ->
                               (command inst, principal, 'd, 'e) Form)
        (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
        (certificatesr3a (npriv :npriv) (privcmd :privcmd)
           (PR privcmd) :('d, 'e) certificate list)
        (SM0StateInterp :state ->
                         (command inst, principal, 'd, 'e) Form)
        (mkinMsg
           (mapSM0r1input
              (mapSM0inputOperatorBob
                 (Name (Role Commander) says
                  (prop (SOME (PR privcmd) :command inst) :
                     (command inst, principal, 'd, 'e) Form))))::
             (ins :inMsg list)) (s :state) (outs :output list)) ==>
   (M,Oi,Os) sat
   (prop (NONE :command inst) :(command inst, principal, 'd, 'e) Form):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***
```

4.3 Proof of SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm

4.3.1 Relevant Code

```
val th1 =
ISPECL
[``MsgInterpret:inMsg -> (command inst, principal, 'd,'e)Form``,
 ``certificateInterpret
   :('d,'e) certificate -> (command inst,principal,'d,'e)Form``,
 ``inputOKr2:(command inst,principal,'d,'e)Form -> bool``,
``(certificatesr3a npriv privcmd (PR privcmd)):('d,'e)certificate list``,
``SM0StateInterp:state -> (command inst,principal,'d,'e)Form``,
``mkinMsg(mapSM0r1input
    (mapSM0inputOperatorBob
     ((Name (Role Commander)) says
      (prop (SOME (PR (privcmd:privcmd)))
      :(command inst,principal,'d,'e)Form))))``,
  ``PR privcmd``,``ins:inMsg list``,``s:state``,``outs:output list``]
TR2_trap_cmd_rule
val SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm =
REWRITE_RULE
[SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma]th1

val _ = save_thm(
   "SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm",
   SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm)
```

4.3.2 Session Transcript

```
> val SMOr3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm                    6
   =
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, principal, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po).
     TR2 (M,Oi,Os) (trap (PR (privcmd :privcmd)))
       (CFG2
          (MsgInterpret :inMsg ->
                        (command inst, principal, 'd, 'e) Form)
          (certificateInterpret :('d, 'e) certificate ->
                                (command inst, principal, 'd, 'e) Form)
          (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (certificatesr3a (npriv :npriv) privcmd (PR privcmd) :
             ('d, 'e) certificate list)
          (SMOStateInterp :state ->
                          (command inst, principal, 'd, 'e) Form)
          (mkinMsg
             (mapSMOr1input
                (mapSMOinputOperatorBob
                   (Name (Role Commander) says
                    (prop (SOME (PR privcmd) :command inst) :
                       (command inst, principal, 'd, 'e) Form))))::
               (ins :inMsg list)) (s :state) (outs :output list))
       (CFG2
          (MsgInterpret :inMsg ->
                        (command inst, principal, 'd, 'e) Form)
          (certificateInterpret :('d, 'e) certificate ->
                                (command inst, principal, 'd, 'e) Form)
          (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (certificatesr3a npriv privcmd (PR privcmd) :
             ('d, 'e) certificate list)
          (SMOStateInterp :state ->
                          (command inst, principal, 'd, 'e) Form) ins
          (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) <=>
     inputOKr2
       (MsgInterpret
          (mkinMsg
             (mapSMOr1input
                (mapSMOinputOperatorBob
                   (Name (Role Commander) says
                    (prop (SOME (PR privcmd) :command inst) :
                       (command inst, principal, 'd, 'e) Form))))) :
          (command inst, principal, 'd, 'e) Form) /\
     CFG2Interpret (M,Oi,Os)
       (CFG2
          (MsgInterpret :inMsg ->
                        (command inst, principal, 'd, 'e) Form)
          (certificateInterpret :('d, 'e) certificate ->
                                (command inst, principal, 'd, 'e) Form)
          (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (certificatesr3a npriv privcmd (PR privcmd) :
             ('d, 'e) certificate list)
          (SMOStateInterp :state ->
                          (command inst, principal, 'd, 'e) Form)
          (mkinMsg
             (mapSMOr1input
                (mapSMOinputOperatorBob
                   (Name (Role Commander) says
                    (prop (SOME (PR privcmd) :command inst) :
                       (command inst, principal, 'd, 'e) Form))))::ins)
          s outs) /\
     (M,Oi,Os) sat
     (prop (NONE :command inst) :(command inst, principal, 'd, 'e) Form):
   thm
val th1 =
   |- (!(M :(command inst, 'b, principal, 'd, 'e) Kripke) (Oi :'d po)
       (Os :'e po).
      CFG2Interpret (M,Oi,Os)
        (CFG2
           (MsgInterpret :inMsg ->
                         (command inst, principal, 'd, 'e) Form)
           (certificateInterpret :('d, 'e) certificate ->
                                 (command inst, principal, 'd, 'e)
                                 Form)
           (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
           (certificatesr3a (npriv :npriv) (privcmd :privcmd)
              (PR privcmd) :('d, 'e) certificate list)
           (SMOStateInterp :state ->
                           (command inst, principal, 'd, 'e) Form)
           (mkinMsg
              (mapSMOr1input
                 (mapSMOinputOperatorBob
                    (Name (Role Commander) says
```

4.4 Proof of SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_thm

4.4.1 Relevant Code

```
val SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_thm =
REWRITE_RULE
[SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma,
 mapSM0r1input_def, mapSM0inputOperatorBob_def,
 mkinMsg_def]
 SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm

val _ = save_thm("SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_thm",
    SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_thm)
```

4.4.2 Session Transcript

7

```
> # # # # # # # # # val SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_thm =
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, principal, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po).
     TR2 (M,Oi,Os) (trap (PR (privcmd :privcmd)))
       (CFG2
          (MsgInterpret :inMsg ->
                        (command inst, principal, 'd, 'e) Form)
          (certificateInterpret :('d, 'e) certificate ->
                                (command inst, principal, 'd, 'e) Form)
          (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (certificatesr3a (npriv :npriv) privcmd (PR privcmd) :
             ('d, 'e) certificate list)
          (SMOStateInterp :state ->
                          (command inst, principal, 'd, 'e) Form)
          (MSG Alice (Order Commander (PR privcmd) :order)
             (sign (privK Alice)
                (hash
                   (SOME (Order Commander (PR privcmd) :order) :
                      order option))::(ins :inMsg list)) (s :state)
          (outs :output list))
       (CFG2
          (MsgInterpret :inMsg ->
                        (command inst, principal, 'd, 'e) Form)
          (certificateInterpret :('d, 'e) certificate ->
                                (command inst, principal, 'd, 'e) Form)
          (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (certificatesr3a npriv privcmd (PR privcmd) :
             ('d, 'e) certificate list)
          (SMOStateInterp :state ->
                          (command inst, principal, 'd, 'e) Form) ins
          (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) <=>
     inputOKr2
       (MsgInterpret
          (MSG Alice (Order Commander (PR privcmd) :order)
             (sign (privK Alice)
                (hash
                   (SOME (Order Commander (PR privcmd) :order) :
                      order option)))) :
          (command inst, principal, 'd, 'e) Form) /\
     CFG2Interpret (M,Oi,Os)
       (CFG2
          (MsgInterpret :inMsg ->
                        (command inst, principal, 'd, 'e) Form)
          (certificateInterpret :('d, 'e) certificate ->
                                (command inst, principal, 'd, 'e) Form)
          (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (certificatesr3a npriv privcmd (PR privcmd) :
             ('d, 'e) certificate list)
          (SMOStateInterp :state ->
                          (command inst, principal, 'd, 'e) Form)
          (MSG Alice (Order Commander (PR privcmd) :order)
             (sign (privK Alice)
                (hash
                   (SOME (Order Commander (PR privcmd) :order) :
                      order option)))::ins) s outs) /\
     (M,Oi,Os) sat
     (prop (NONE :command inst) :(command inst, principal, 'd, 'e) Form):
   thm
>
*** Emacs/HOL command completed ***
```

4.5 Proof of SM0r3_Alice_TR2_iff_TR_trap_privcmd

4.5.1 Relevant Code

```
val th1 =
ISPECL
```

```
[``MsgInterpret:inMsg -> (command inst, principal, 'd,'e)Form``,
 ``certificateInterpret
   :('d,'e) certificate -> (command inst,principal,'d,'e)Form``,
 ``inputOKr2:(command inst,principal,'d,'e)Form -> bool``,
 ``(certsr2a npriv privcmd (PR privcmd)):
    (command inst,principal,'d,'e)Form list``,
 ``(certificatesr3a npriv privcmd(PR privcmd)):('d,'e)certificate list``,
 ``SM0StateInterp:state -> (command inst,principal,'d,'e)Form``,
 ``mkinMsg(mapSM0r1input
    (mapSM0inputOperatorBob
     ((Name (Role Commander)) says
      (prop (SOME (PR (privcmd:privcmd)))
      :(command inst,principal,'d,'e)Form))))``,
  ``(Name (KeyS (pubK Alice))) quoting (Name (Role Commander))``,
  ``PR privcmd``,``ins:(command inst, principal,'d,'e)Form list``,
  ``ins2:inMsg list``,``s:state``,``outs:output list``]
TR2_iff_TR_trap_thm

val th2 =
 REWRITE_RULE[SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma] th1

val th3 =
TAC_PROOF(([],
``(((mapSM0r1input
    (mapSM0inputOperatorBob
     (Name (Role Commander) says prop (SOME (PR privcmd)))))
   :(command inst,principal,'d,'e)Form)::ins) =
 ((MsgInterpret
  (mkinMsg
   ((mapSM0r1input
    (mapSM0inputOperatorBob
     ((Name (Role Commander) says ((prop (SOME (PR privcmd))):
       (command inst,principal,'d,'e)Form)))))))::
         (ins:(command inst,principal,'d,'e)Form list))``),
REWRITE_TAC[mapSM0inputOperatorBob_def,mapSM0r1input_def,
            MsgInterpret_inverts_mkinMsg_thm])

val th4 =
REWRITE_RULE[th3]SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma

val th5 =
REWRITE_RULE[th4]th2

val th6 =
TAC_PROOF(([],
``((MsgInterpret:inMsg -> (command inst,principal,'d,'e)Form)
    (mkinMsg
      (mapSM0r1input
        (mapSM0inputOperatorBob
          (Name (Role Commander) says
           (prop (SOME (PR privcmd))):(command inst,principal,'d,'e)Form)))))) =
   ((Name (KeyS (pubK Alice)) quoting Name (Role Commander) says
    prop (SOME (PR privcmd))):(command inst,principal,'d,'e)Form)``),
REWRITE_TAC[mapSM0inputOperatorBob_def,mapSM0r1input_def,
            MsgInterpret_inverts_mkinMsg_thm])

val th7 =
REWRITE_RULE[th6]th5
```

```
val SM0r3_Alice_TR2_iff_TR_trap_privcmd =
REWRITE_RULE[certificatesr3a_certsr2a_map_thm]th7

val _ = save_thm("SM0r3_Alice_TR2_iff_TR_trap_privcmd",
        SM0r3_Alice_TR2_iff_TR_trap_privcmd)
```

4.5.2 Session Transcript

8

```
> val SMOr3_Alice_TR2_iff_TR_trap_privcmd =
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, principal, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po).
     TR2 (M,Oi,Os) (trap (PR (privcmd :privcmd)))
       (CFG2
          (MsgInterpret :inMsg ->
                        (command inst, principal, 'd, 'e) Form)
          (certificateInterpret :('d, 'e) certificate ->
                                (command inst, principal, 'd, 'e) Form)
          (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (certificatesr3a (npriv :npriv) privcmd (PR privcmd) :
             ('d, 'e) certificate list)
          (SMOStateInterp :state ->
                          (command inst, principal, 'd, 'e) Form)
          (mkinMsg
             (mapSMOr1input
                (mapSMOinputOperatorBob
                   (Name (Role Commander) says
                    (prop (SOME (PR privcmd) :command inst) :
                       (command inst, principal, 'd, 'e) Form))))::
               (ins2 :inMsg list)) (s :state) (outs :output list))
       (CFG2
          (MsgInterpret :inMsg ->
                        (command inst, principal, 'd, 'e) Form)
          (certificateInterpret :('d, 'e) certificate ->
                                (command inst, principal, 'd, 'e) Form)
          (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (certificatesr3a npriv privcmd (PR privcmd) :
             ('d, 'e) certificate list)
          (SMOStateInterp :state ->
                          (command inst, principal, 'd, 'e) Form) ins2
          (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) <=>
      TR (M,Oi,Os) (trap (PR privcmd))
        (CFG (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (SMOStateInterp :state ->
                          (command inst, principal, 'd, 'e) Form)
          (certsr2a npriv privcmd (PR privcmd) :
             (command inst, principal, 'd, 'e) Form list)
          (Name (KeyS (pubK Alice)) quoting Name (Role Commander) says
           (prop (SOME (PR privcmd) :command inst) :
              (command inst, principal, 'd, 'e) Form)::
            (ins :(command inst, principal, 'd, 'e) Form list)) s
          outs)
        (CFG (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
          (SMOStateInterp :state ->
                          (command inst, principal, 'd, 'e) Form)
          (certsr2a npriv privcmd (PR privcmd) :
             (command inst, principal, 'd, 'e) Form list) ins
          (NS s (trap (PR privcmd))) (Out s (trap (PR privcmd))::outs)):
   thm
val th1 =
   |- ((MsgInterpret
       (mkinMsg
          (mapSMOr1input
             (mapSMOinputOperatorBob
                (Name (Role Commander) says
                 (prop (SOME (PR (privcmd :privcmd)) :command inst) :
                    (command inst, principal, 'd, 'e) Form))))) :
       (command inst, principal, 'd, 'e) Form) =
    Name (KeyS (pubK Alice)) quoting Name (Role Commander) says
    (prop (SOME (PR privcmd) :command inst) :
       (command inst, principal, 'd, 'e) Form)) ==>
   (MAP
      (certificateInterpret :('d, 'e) certificate ->
                            (command inst, principal, 'd, 'e) Form)
      (certificatesr3a (npriv :npriv) privcmd (PR privcmd) :
         ('d, 'e) certificate list) =
    (certsr2a npriv privcmd (PR privcmd) :
       (command inst, principal, 'd, 'e) Form list)) ==>
   (!(M :(command inst, 'b, principal, 'd, 'e) Kripke) (Oi :'d po)
       (Os :'e po).
      CFGInterpret (M,Oi,Os)
        (CFG (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
           (SMOStateInterp :state ->
                           (command inst, principal, 'd, 'e) Form)
           (certsr2a npriv privcmd (PR privcmd) :
              (command inst, principal, 'd, 'e) Form list)
           ((MsgInterpret
               (mkinMsg
                  (mapSMOr1input
                     (mapSMOinputOperatorBob
                        (Name (Role Commander) says
```

Appendix A

Source Code for SM0r3Solutions.sml

```
(*****************************************************************************)
(* SM0r3Solutions: Proof that Alice's request to execute a privcmd is trapped *)
(* Author: Shiu-Kai Chin                                                     *)
(* Date: 30 April 2017                                                       *)
(*****************************************************************************)
structure SM0r3SolutionsScript = struct

(* ==== interactive mode ====
app load ["principalTheory","m0TypesTheory","ssm1Theory","ssm2Theory","TypeBase",
          "SM0Theory","SM0r1Theory","SM0r2Theory","SM0r3Theory",
          "aclrulesTheory","aclDrulesTheory","acl_infRules","satListTheory"];
open principalTheory m0TypesTheory ssm1Theory ssm2Theory TypeBase SM0Theory
     SM0r1Theory SM0r2Theory SM0r3Theory certStructureTheory inMsgTheory
     listTheory SM0r3SolutionsTheory;
open aclrulesTheory aclDrulesTheory acl_infRules satListTheory;
 ==== end interactive mode ==== *)

open HolKernel Parse boolLib bossLib;
open ssm1Theory ssm2Theory listTheory;
open SM0r3Theory SM0r2Theory SM0r1Theory SM0Theory principalTheory;
open  certStructureTheory inMsgTheory;
open aclrulesTheory aclDrulesTheory acl_infRules satListTheory;


val _ = new_theory "SM0r3Solutions";

(* ------------------------------------------------------------------------- *)
(* Modify certsr1a to be more general with respect to commands.              *)
(* ------------------------------------------------------------------------- *)
val certsr1a_def =
Define
`certsr1a(npriv:npriv)(privcmd:privcmd)(cmd:command) =
 (certs (npriv:npriv)(privcmd:privcmd)) ++
 [(reps (Name (Staff Alice))(Name (Role Commander))
        ((prop (SOME (cmd:command))):(command inst, principal,'d,'e)Form));
  (reps (Name (Staff Bob))(Name (Role Operator))
        ((prop (SOME (cmd:command))):(command inst, principal,'d,'e)Form))]`

(*****************************************************************************)
(* Define certsr2a in terms of certsr1a, certsr2root, and certsr2signed      *)
(*****************************************************************************)
val certsr2a_def =
Define
`certsr2a (npriv:npriv)(privcmd:privcmd)(cmd:command) =
 (certsr1a (npriv:npriv)(privcmd:privcmd)(cmd:command) ++
 (certsr2root npriv privcmd))
 ++ (certsr2signed npriv privcmd)`
```

```
(*****************************************************************************)
(* Create the list of root and signed certificates corresponding to certs,   *)
(* certsr1, and certsr2.                                                     *)
(*****************************************************************************)
val certificatesr3a_def =
Define
`certificatesr3a (npriv:npriv) (privcmd:privcmd) (cmd:command) =
 (MAP mkRCert ((certsr1a npriv privcmd cmd) ++ (certsr2root npriv privcmd))) ++
 (MAP (mkSCert (ca 0))(certsr2signed npriv privcmd))`


(* ==== start here ====*)


(* ========================================================================== *)
(* The following HOL terms correspond to the theorems you are to prove as     *)
(* part of your projects.  You can prove the theorems using any proof style   *)
(* you would like, i.e., forward, goal oriented, or a combination of both.    *)
(* ========================================================================== *)


(*****************************************************************************)
(* Proof of SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma                 *)
(*****************************************************************************)
val th1 =
TAC_PROOF(([],
``((M,Oi,Os) satList (certsr2a npriv privcmd (PR privcmd))) ==>
  (((M:(command inst,'b, principal,'d,'e)Kripke),Oi,Os)
  sat Name (KeyS (pubK Alice)) speaks_for Name (Staff Alice))``),
REWRITE_TAC[certsr2a_def,certsr2root_def,certsr2signed_def,
            certsr1a_def,certs_def,satList_CONS,
            (GSYM satList_conj),satList_nil] THEN
PROVE_TAC[Derived_Speaks_For,Controls])

val th2 =
TAC_PROOF(
([],``(((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
        sat (Name (Role Commander)) speaks_for (Name (Role Commander)))``),
PROVE_TAC[Idemp_Speaks_For])

val th3 =
TAC_PROOF(
([],``(((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
       satList (certsr2a npriv privcmd (PR privcmd))) ==>
      ((M,Oi,Os) sat (((Name (KeyS (pubK Alice))) quoting
                       (Name (Role Commander)))
                      speaks_for
                     ((Name (Staff Alice))
                     quoting (Name (Role Commander)))))``),
PROVE_TAC[Mono_speaks_for,th1,Derived_Speaks_For,th2])

val th4 =
TAC_PROOF(
([],``(((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
       satList (certsr2a npriv privcmd(PR privcmd))) ==>
((M,Oi,Os) sat ((((Name (KeyS (pubK Alice)))
quoting (Name (Role Commander))) says
```

```
        (prop (SOME (PR privcmd)))):
        (command inst,principal,'d,'e)Form)) ==>
       ((M,Oi,Os) sat ((((Name (Staff Alice))
       quoting (Name (Role Commander))) says
        (prop (SOME (PR privcmd)))):
        (command inst,principal,'d,'e)Form))``),
PROVE_TAC[th3,Derived_Speaks_For])



val th5 =
TAC_PROOF(
([],``(((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
        satList (certsr2a npriv privcmd (PR privcmd))) ==>
((M,Oi,Os) sat (Name (Role Commander) says prop
(SOME (PR privcmd))) impf (prop NONE)) /\
       ((M,Oi,Os) sat (reps (Name (Staff Alice)) (Name (Role Commander))
        (prop (SOME (PR privcmd)))))``),
REWRITE_TAC[certsr2a_def,certsr2root_def,certsr2signed_def,certsr1a_def,
            certs_def,satList_CONS,satList_nil,(GSYM satList_conj)] THEN
PROVE_TAC[])


val th6 =
TAC_PROOF(([],
``(((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os) satList
   (certsr2a npriv privcmd (PR privcmd))) ==>
    ((M,Oi,Os) sat ((((Name (KeyS (pubK Alice)))
    quoting (Name (Role Commander))) says
        (prop (SOME (PR privcmd)))):(command inst,principal,'d,'e)Form))
  ==> ((M,Oi,Os) sat (prop NONE))``),
PROVE_TAC[Rep_Says,th4,th5,Modus_Ponens])

val SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma =
TAC_PROOF(([],
``CFGInterpret ((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
  (CFG inputOKr2 SM0StateInterp (certsr2a npriv privcmd (PR privcmd))
   (mapSM0r1input
    (mapSM0inputOperatorBob
     ((Name (Role Commander)) says (prop (SOME (PR (privcmd:privcmd))))))::ins)
   s (outs:output list)) ==>
  ((M,Oi,Os) sat (prop NONE))``),
REWRITE_TAC[CFGInterpret_def,mapSM0inputOperatorBob_def,mapSM0r1input_def] THEN
PROVE_TAC[th6])

val _ = save_thm("SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma",
SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma);


(*****************************************************************************)
(* Proof of SM0r2_Commander_Alice_trap_privcmd_justified_thm                 *)
(*****************************************************************************)

val SM0r2_Commander_Alice_trap_privcmd_justified_thm =
let
 val th1 =
  ISPECL
 [``inputOKr2:(command inst, principal,'d,'e)Form -> bool``,
```

```
 ``SM0StateInterp:state->(command inst, principal,'d,'e)Form``,
``(certsr2a npriv privcmd (PR privcmd)):
  (command inst, principal,'d,'e)Form list``,
``(Name (KeyS (pubK Alice))) quoting (Name (Role Commander))``,``PR privcmd``,
          ``ins:(command inst,principal,'d,'e)Form list``,
            ``s:state``,``outs:output list``]
 TR_trap_cmd_rule
 val th2 =
 REWRITE_RULE[mapSM0inputOperatorBob_def,mapSM0r1input_def]
 SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma
in
   REWRITE_RULE[th2]th1
end

val _ = save_thm("SM0r2_Commander_Alice_trap_privcmd_justified_thm",
        SM0r2_Commander_Alice_trap_privcmd_justified_thm)

(*****************************************************************************)
(* Proof of SM0r2_Commander_mapSM0r1input_trap_privcmd_justifed_thm           *)
(*****************************************************************************)
val SM0r2_Commander_mapSM0r1input_trap_privcmd_justifed_thm =
let
 val th1 =
  TAC_PROOF(([],
    ``((Name (KeyS (pubK Alice))) quoting (Name (Role Commander))
       says (prop (SOME (PR (privcmd:privcmd))))
            :(command inst, principal,'d,'e)Form)
       =
       mapSM0r1input
          (mapSM0inputOperatorBob
          ((Name (Role Commander)) says (prop (SOME (PR (privcmd:privcmd)))))
                    :(command inst, principal,'d,'e)Form)``),
  PROVE_TAC[mapSM0r1input_def,mapSM0inputOperatorBob_def])
in
 REWRITE_RULE[th1]SM0r2_Commander_Alice_trap_privcmd_justified_thm
end

val _ = save_thm("SM0r2_Commander_mapSM0r1input_trap_privcmd_justifed_thm",
        SM0r2_Commander_mapSM0r1input_trap_privcmd_justifed_thm)


(*****************************************************************************)
(* Proof of certificatesr3a_certsr2a_map_thm                                 *)
(*****************************************************************************)
val certificatesr3a_certsr2a_map_thm =
TAC_PROOF(([],
``!(npriv :npriv) (privcmd :privcmd).
    MAP
      (certificateInterpret :('a, 'b) certificate ->
                             (command inst, principal, 'a, 'b) Form)
      (certificatesr3a npriv privcmd (PR privcmd) :
         ('a, 'b) certificate list) =
    (certsr2a npriv privcmd (PR privcmd) :
       (command inst, principal, 'a, 'b) Form list)``),
REWRITE_TAC[certificatesr3a_def,MAP_APPEND] THEN
REWRITE_TAC[MAP_certificateInterpret_mkSCert_thm,
 MAP_certificateInterpret_mkRCert_thm,certsr2a_def])
```

```
val _ = save_thm ("certificatesr3a_certsr2a_map_thm",
    certificatesr3a_certsr2a_map_thm)


(*****************************************************************************)
(* Proof of SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma            *)
(*****************************************************************************)

val SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma =
TAC_PROOF(([],
``CFG2Interpret
    ((M :(command inst, 'b, principal, 'd, 'e) Kripke),(Oi :'d po),
     (Os :'e po))
    (CFG2
       (MsgInterpret :inMsg -> (command inst, principal, 'd, 'e) Form)
       (certificateInterpret :('d, 'e) certificate ->
                              (command inst, principal, 'd, 'e) Form)
       (inputOKr2 :(command inst, principal, 'd, 'e) Form -> bool)
       (certificatesr3a (npriv :npriv) (privcmd :privcmd) (PR privcmd) :
          ('d, 'e) certificate list)
       (SM0StateInterp :state -> (command inst, principal, 'd, 'e) Form)
       (mkinMsg
          (mapSM0r1input
             (mapSM0inputOperatorBob
                (Name (Role Commander) says
                 (prop (SOME (PR privcmd) :command inst) :
                    (command inst, principal, 'd, 'e) Form))))::
            (ins :inMsg list)) (s :state) (outs :output list)) ==>
  (M,Oi,Os) sat
  (prop (NONE :command inst) :(command inst, principal, 'd, 'e) Form)``),
  REWRITE_TAC
 [CFG2Interpret_def,mapSM0inputOperatorBob_def,mapSM0r1input_def,
  MsgInterpret_inverts_mkinMsg_thm,
  certificatesr3a_certsr2a_map_thm] THEN
PROVE_TAC[th6])


val _ = save_thm("SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma",
        SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma)



(*****************************************************************************)
(* Proof of                                                                  *)
(* SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm *)
(*****************************************************************************)


val th1 =
ISPECL
[``MsgInterpret:inMsg -> (command inst, principal, 'd,'e)Form``,
 ``certificateInterpret
   :('d,'e) certificate -> (command inst,principal,'d,'e)Form``,
 ``inputOKr2:(command inst,principal,'d,'e)Form -> bool``,
``(certificatesr3a npriv privcmd (PR privcmd)):('d,'e)certificate list``,
``SM0StateInterp:state -> (command inst,principal,'d,'e)Form``,
``mkinMsg(mapSM0r1input
    (mapSM0inputOperatorBob
```

```
      ((Name (Role Commander)) says
       (prop (SOME (PR (privcmd:privcmd)))
       :(command inst,principal,'d,'e)Form))))``,
  ``PR privcmd``,``ins:inMsg list``,``s:state``,``outs:output list``]
TR2_trap_cmd_rule
val SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm =
REWRITE_RULE
[SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma]th1

val _ = save_thm(
   "SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm",
   SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm)

(*****************************************************************************)
(* Proof of SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_thm          *)
(*****************************************************************************)


val SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_thm =
REWRITE_RULE
[SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma,
 mapSM0r1input_def,mapSM0inputOperatorBob_def,
 mkinMsg_def]
 SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_with_refinements_thm

val _ = save_thm("SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_thm",
    SM0r3_Commander_Alice_privcmd_trap_privcmd_justified_thm)


(*****************************************************************************)
(* Proof of  SM0r3_Alice_TR2_iff_TR_trap_privcmd                             *)
(*****************************************************************************)

val th1 =
ISPECL
[``MsgInterpret:inMsg -> (command inst, principal, 'd,'e)Form``,
 ``certificateInterpret
   :('d,'e) certificate -> (command inst,principal,'d,'e)Form``,
 ``inputOKr2:(command inst,principal,'d,'e)Form -> bool``,
 ``(certsr2a npriv privcmd (PR privcmd)):
    (command inst,principal,'d,'e)Form list``,
 ``(certificatesr3a npriv privcmd(PR privcmd)):('d,'e)certificate list``,
 ``SM0StateInterp:state -> (command inst,principal,'d,'e)Form``,
 ``mkinMsg(mapSM0r1input
    (mapSM0inputOperatorBob
     ((Name (Role Commander)) says
      (prop (SOME (PR (privcmd:privcmd)))
      :(command inst,principal,'d,'e)Form))))``,
  ``(Name (KeyS (pubK Alice))) quoting (Name (Role Commander))``,
  ``PR privcmd``,``ins:(command inst, principal,'d,'e)Form list``,
  ``ins2:inMsg list``,``s:state``,``outs:output list``]
TR2_iff_TR_trap_thm

val th2 =
 REWRITE_RULE[SM0r3_mkinMsg_SM0r2_Alice_Commander_trap_privcmd_lemma] th1

val th3 =
TAC_PROOF(([],
```

```
``(((mapSM0r1input
     (mapSM0inputOperatorBob
      (Name (Role Commander) says prop (SOME (PR privcmd)))))
   :(command inst,principal,'d,'e)Form)::ins) =
 ((MsgInterpret
  (mkinMsg
   ((mapSM0r1input
     (mapSM0inputOperatorBob
      ((Name (Role Commander) says ((prop (SOME (PR privcmd))):
        (command inst,principal,'d,'e)Form)))))))::
          (ins:(command inst,principal,'d,'e)Form list))``),
REWRITE_TAC[mapSM0inputOperatorBob_def,mapSM0r1input_def,
            MsgInterpret_inverts_mkinMsg_thm])

val th4 =
REWRITE_RULE[th3]SM0r2_mapSM0r1_Alice_Commander_trap_privcmd_lemma

val th5 =
REWRITE_RULE[th4]th2

val th6 =
TAC_PROOF(([],
``((MsgInterpret:inMsg -> (command inst,principal,'d,'e)Form)
     (mkinMsg
       (mapSM0r1input
         (mapSM0inputOperatorBob
           (Name (Role Commander) says
            (prop (SOME (PR privcmd))):(command inst,principal,'d,'e)Form)))) =
   ((Name (KeyS (pubK Alice)) quoting Name (Role Commander) says
     prop (SOME (PR privcmd))):(command inst,principal,'d,'e)Form)``),
REWRITE_TAC[mapSM0inputOperatorBob_def,mapSM0r1input_def,
            MsgInterpret_inverts_mkinMsg_thm])

val th7 =
REWRITE_RULE[th6]th5

val SM0r3_Alice_TR2_iff_TR_trap_privcmd =
REWRITE_RULE[certificatesr3a_certsr2a_map_thm]th7

val _ = save_thm("SM0r3_Alice_TR2_iff_TR_trap_privcmd",
        SM0r3_Alice_TR2_iff_TR_trap_privcmd)

(*==== end here ==== *)

val _ = export_theory();



end (* structure *)
```