Homework 12

Chirag Sachdev

Week 12

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:

[SMO_Commander_privcmd_trapped_lemma]

```
 ⊢ CFGInterpret (M,Oi,Os)
     (CFG inputOK SMOStateInterp (certs npriv privcmd)
        (Name (Role Commander) says prop (SOME (PR privcmd))::
              ins) s outs) ⇒
   (M,Oi,Os) sat prop NONE
```

[SMO_Commander_trap_privcmd_justified_thm]

```
 ⊢ ∀NS Out M Oi Os.
     TR (M,Oi,Os) (trap (PR privcmd))
       (CFG inputOK SMOStateInterp (certs npriv privcmd)
          (Name (Role Commander) says prop (SOME (PR privcmd))::
               ins) s outs)
       (CFG inputOK SMOStateInterp (certs npriv privcmd) ins
          (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) ⟺
     inputOK
       (Name (Role Commander) says prop (SOME (PR privcmd))) ∧
     CFGInterpret (M,Oi,Os)
       (CFG inputOK SMOStateInterp (certs npriv privcmd)
          (Name (Role Commander) says prop (SOME (PR privcmd))::
               ins) s outs) ∧ (M,Oi,Os) sat prop NONE
```

[SMOr1_mapSMO_Alice_Commander_trap_privcmd_lemma]

```
 ⊢ CFGInterpret (M,Oi,Os)
     (CFG inputOKr1 SMOStateInterp
        (certsr1a npriv privcmd (PR privcmd))
        (mapSMOinputOperatorBob
           (Name (Role Commander) says
            prop (SOME (PR privcmd)))::ins) s outs) ⇒
   (M,Oi,Os) sat prop NONE
```

[SMOr1_Commander_Alice_trap_privcmd_justified_thm]

```
 ⊢ ∀NS Out M Oi Os.
     TR (M,Oi,Os) (trap (PR privcmd))
       (CFG inputOKr1 SMOStateInterp
          (certsr1a npriv privcmd (PR privcmd))
          (Name (Staff Alice) quoting Name (Role Commander) says
           prop (SOME (PR privcmd))::ins) s outs)
       (CFG inputOKr1 SMOStateInterp
          (certsr1a npriv privcmd (PR privcmd)) ins
          (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) ⟺
     inputOKr1
       (Name (Staff Alice) quoting Name (Role Commander) says
        prop (SOME (PR privcmd))) ∧
     CFGInterpret (M,Oi,Os)
       (CFG inputOKr1 SMOStateInterp
          (certsr1a npriv privcmd (PR privcmd))
          (Name (Staff Alice) quoting Name (Role Commander) says
           prop (SOME (PR privcmd))::ins) s outs) ∧
     (M,Oi,Os) sat prop NONE
```

[SMOr1_Commander_mapSMOinputOperatorBob_trap_privcmd_justified_thm]

```
 ⊢ ∀s privcmd outs npriv ins NS Out M Oi Os.
     TR (M,Oi,Os) (trap (PR privcmd))
       (CFG inputOKr1 SMOStateInterp
          (certsr1a npriv privcmd (PR privcmd))
          (mapSMOinputOperatorBob
             (Name (Role Commander) says
              prop (SOME (PR privcmd)))::ins) s outs)
       (CFG inputOKr1 SMOStateInterp
          (certsr1a npriv privcmd (PR privcmd)) ins
          (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) ⟺
     inputOKr1
       (mapSMOinputOperatorBob
          (Name (Role Commander) says
           prop (SOME (PR privcmd)))) ∧
     CFGInterpret (M,Oi,Os)
       (CFG inputOKr1 SMOStateInterp
          (certsr1a npriv privcmd (PR privcmd))
          (mapSMOinputOperatorBob
             (Name (Role Commander) says
              prop (SOME (PR privcmd)))::ins) s outs) ∧
     (M,Oi,Os) sat prop NONE
```

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

Chapter 2

Problem statement

Prove the following theorems:

[SM0_Commander_privcmd_trapped_lemma]

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

[SM0_Commander_trap_privcmd_justified_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOK SM0StateInterp (certs npriv privcmd)
         (Name (Role Commander) says prop (SOME (PR privcmd))::
              ins) s outs)
      (CFG inputOK SM0StateInterp (certs npriv privcmd) ins
         (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⇔
    inputOK
      (Name (Role Commander) says prop (SOME (PR privcmd))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK SM0StateInterp (certs npriv privcmd)
         (Name (Role Commander) says prop (SOME (PR privcmd))::
              ins) s outs) ∧ (M,Oi,Os) sat prop NONE
```

[SM0r1_mapSM0_Alice_Commander_trap_privcmd_lemma]

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG inputOKr1 SM0StateInterp
       (certsr1a npriv privcmd (PR privcmd))
       (mapSM0inputOperatorBob
          (Name (Role Commander) says
           prop (SOME (PR privcmd)))::ins) s outs) ⇒
  (M,Oi,Os) sat prop NONE
```

[SM0r1_Commander_Alice_trap_privcmd_justified_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOKr1 SM0StateInterp
         (certsr1a npriv privcmd (PR privcmd))
         (Name (Staff Alice) quoting Name (Role Commander) says
          prop (SOME (PR privcmd))::ins) s outs)
      (CFG inputOKr1 SM0StateInterp
         (certsr1a npriv privcmd (PR privcmd)) ins
         (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⇔
    inputOKr1
      (Name (Staff Alice) quoting Name (Role Commander) says
      prop (SOME (PR privcmd))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOKr1 SMOStateInterp
         (certsr1a npriv privcmd (PR privcmd))
         (Name (Staff Alice) quoting Name (Role Commander) says
          prop (SOME (PR privcmd))::ins) s outs) ∧
    (M,Oi,Os) sat prop NONE
```

[SMOr1_Commander_mapSMOinputOperatorBob_trap_privcmd_justified_thm]

```
 ⊢ ∀s privcmd outs npriv ins NS Out M Oi Os.
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOKr1 SMOStateInterp
         (certsr1a npriv privcmd (PR privcmd))
         (mapSMOinputOperatorBob
            (Name (Role Commander) says
             prop (SOME (PR privcmd)))::ins) s outs)
      (CFG inputOKr1 SMOStateInterp
         (certsr1a npriv privcmd (PR privcmd)) ins
         (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⟺
    inputOKr1
      (mapSMOinputOperatorBob
         (Name (Role Commander) says
          prop (SOME (PR privcmd)))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOKr1 SMOStateInterp
         (certsr1a npriv privcmd (PR privcmd))
         (mapSMOinputOperatorBob
            (Name (Role Commander) says
             prop (SOME (PR privcmd)))::ins) s outs) ∧
    (M,Oi,Os) sat prop NONE
```

Chapter 3

SM0 Solutions with the following sections

3.1 Proof of SMO_Commander_privcmd_trapped_lemma

3.1.1 Relevant Code

```
val SM0_Commander_privcmd_trapped_lemma =
TAC_PROOF(([],
``CFGInterpret ((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
  (CFG inputOK SM0StateInterp (certs npriv privcmd)
   (((Name (Role Commander)) says (prop (SOME (PR (privcmd:privcmd)))))::ins)
   s (outs:output list)) ==>
  ((M,Oi,Os) sat (prop NONE))``),
REWRITE_TAC[CFGInterpret_def,certs_def,SM0StateInterp_def,satList_CONS,
            satList_nil,sat_TT] THEN
PROVE_TAC[Controls, Modus_Ponens])

val _ = save_thm("SM0_Commander_privcmd_trapped_lemma",
SM0_Commander_privcmd_trapped_lemma)
```

3.1.2 Session Transcript

```
Meson search level: ....                                                   1
val SM0_Commander_privcmd_trapped_lemma =
   |- CFGInterpret
     ((M :(command inst, 'b, principal, 'd, 'e) Kripke),(Oi :'d po),
      (Os :'e po))
     (CFG (inputOK :(command inst, principal, 'd, 'e) Form -> bool)
        (SM0StateInterp :state ->
                         (command inst, principal, 'd, 'e) Form)
        (certs (npriv :npriv) (privcmd :privcmd) :
           (command inst, principal, 'd, 'e) Form list)
        (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 it = (): unit
>
*** Emacs/HOL command completed ***
```

3.2 Proof of SM0_Commander_trap_privcmd_justified_thm

3.2.1 Relevant Code

```
val SM0_Commander_trap_privcmd_justified_thm =
let
 val th1 =
 ISPECL
 [``inputOK:(command inst, principal,'d,'e)Form -> bool``,
  ``SM0StateInterp:state->(command inst, principal,'d,'e)Form``,
  ``(certs npriv privcmd):(command inst, principal,'d,'e)Form list``,
  ``Name (Role Commander)``,``PR privcmd``,
  ``ins:(command inst,principal,'d,'e)Form list``,
  ``s:state``,``outs:output list``]
 TR_trap_cmd_rule
in
 TAC_PROOF(([],
``!(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 (inputOK :(command inst, principal, 'd, 'e) Form -> bool)
        (SM0StateInterp :state ->
                         (command inst, principal, 'd, 'e) Form)
        (certs (npriv :npriv) privcmd :
           (command inst, principal, 'd, 'e) Form list)
        (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 (inputOK :(command inst, principal, 'd, 'e) Form -> bool)
        (SM0StateInterp :state ->
                         (command inst, principal, 'd, 'e) Form)
        (certs npriv privcmd :
           (command inst, principal, 'd, 'e) Form list) ins
        (NS s (trap (PR privcmd)))
        (Out s (trap (PR privcmd))::outs)) <=>
   inputOK
     (Name (Role Commander) says
      (prop (SOME (PR privcmd) :command inst) :
         (command inst, principal, 'd, 'e) Form)) /\
   CFGInterpret (M,Oi,Os)
     (CFG (inputOK :(command inst, principal, 'd, 'e) Form -> bool)
        (SM0StateInterp :state ->
                         (command inst, principal, 'd, 'e) Form)
        (certs npriv privcmd :
           (command inst, principal, 'd, 'e) Form list)
        (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)``),
 PROVE_TAC[th1,SM0_Commander_privcmd_trapped_lemma])
```

```
end

val _ = save_thm ("SM0_Commander_trap_privcmd_justified_thm",
SM0_Commander_trap_privcmd_justified_thm)
```

3.2.2 Session Transcript

2

```
> Meson search level: .....................................
val SM0_Commander_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).
     TR (M,Oi,Os) (trap (PR (privcmd :privcmd)))
       (CFG (inputOK :(command inst, principal, 'd, 'e) Form -> bool)
          (SM0StateInterp :state ->
                           (command inst, principal, 'd, 'e) Form)
          (certs (npriv :npriv) privcmd :
             (command inst, principal, 'd, 'e) Form list)
          (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 (inputOK :(command inst, principal, 'd, 'e) Form -> bool)
          (SM0StateInterp :state ->
                           (command inst, principal, 'd, 'e) Form)
          (certs npriv privcmd :
             (command inst, principal, 'd, 'e) Form list) ins
          (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) <=>
     inputOK
       (Name (Role Commander) says
        (prop (SOME (PR privcmd) :command inst) :
           (command inst, principal, 'd, 'e) Form)) /\
     CFGInterpret (M,Oi,Os)
       (CFG (inputOK :(command inst, principal, 'd, 'e) Form -> bool)
          (SM0StateInterp :state ->
                           (command inst, principal, 'd, 'e) Form)
          (certs npriv privcmd :
             (command inst, principal, 'd, 'e) Form list)
          (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

SM0r1 Solutions with the following sections

4.1 Proof of SM0r1_mapSM0_Alice_Commander_trap_privcmd_lemma

4.1.1 Relevant Code

```
val th1=
TAC_PROOF(([],
``(((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os) satList
    (certsr1a npriv privcmd (PR privcmd))) ==>
   ((M,Oi,Os) sat ((Name (Staff Alice)) quoting (Name (Role Commander)) says
   (prop (SOME (PR (privcmd:privcmd))))))
   ==> ((M,Oi,Os) sat (prop NONE))``),
REWRITE_TAC
  [certsr1a_def,certs_def,satList_CONS,satList_nil,(GSYM satList_conj)] THEN
PROVE_TAC[Rep_Says, Modus_Ponens])

val SM0r1_mapSM0_Alice_Commander_trap_privcmd_lemma =
TAC_PROOF(([],
``CFGInterpret ((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
   (CFG inputOKr1 SM0StateInterp (certsr1a npriv privcmd (PR privcmd))
    (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] THEN
PROVE_TAC[th1])

val _ = save_thm("SM0r1_mapSM0_Alice_Commander_trap_privcmd_lemma",
SM0r1_mapSM0_Alice_Commander_trap_privcmd_lemma)
```

4.1.2 Session Transcript

```
> Meson search level: ......                                                    3
Meson search level: ....
val SM0r1_mapSM0_Alice_Commander_trap_privcmd_lemma =
   |- CFGInterpret
     ((M :(command inst, 'b, principal, 'd, 'e) Kripke),(Oi :'d po),
      (Os :'e po))
     (CFG (inputOKr1 :(command inst, principal, 'd, 'e) Form -> bool)
        (SM0StateInterp :state ->
                         (command inst, principal, 'd, 'e) Form)
        (certsr1a (npriv :npriv) (privcmd :privcmd) (PR privcmd) :
           (command inst, principal, 'd, 'e) Form list)
        (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)) ==>
   (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
   (certsr1a (npriv :npriv) (privcmd :privcmd) (PR privcmd) :
      (command inst, principal, 'd, 'e) Form list) ==>
   (M,Oi,Os) sat
   Name (Staff Alice) quoting Name (Role Commander) says
   (prop (SOME (PR privcmd) :command inst) :
      (command inst, principal, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   (prop (NONE :command inst) :(command inst, principal, 'd, 'e) Form):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***
```

4.2 Proof of SM0r1_Commander_Alice_trap_privcmd_justified_thm

4.2.1 Relevant Code

```
val SM0r1_Commander_Alice_trap_privcmd_justified_thm =
let
 val th1 =
 ISPECL
 [``inputOKr1:(command inst, principal,'d,'e)Form -> bool``,
  ``SM0StateInterp:state->(command inst, principal,'d,'e)Form``,
  ``(certsr1a npriv privcmd (PR privcmd)):(command inst, principal,'d,'e)Form list
     ``,
  ``(Name (Staff 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]
     SM0r1_mapSM0_Alice_Commander_trap_privcmd_lemma
in
 REWRITE_RULE[th2]th1
end

val _ = save_thm ("SM0r1_Commander_Alice_trap_privcmd_justified_thm",
 SM0r1_Commander_Alice_trap_privcmd_justified_thm)
```

4.2.2 Session Transcript

4

```
> val SMOr1_Commander_Alice_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).
     TR (M,Oi,Os) (trap (PR (privcmd :privcmd)))
       (CFG (inputOKr1 :(command inst, principal, 'd, 'e) Form -> bool)
          (SMOStateInterp :state ->
                           (command inst, principal, 'd, 'e) Form)
          (certsr1a (npriv :npriv) privcmd (PR privcmd) :
             (command inst, principal, 'd, 'e) Form list)
          (Name (Staff 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 (inputOKr1 :(command inst, principal, 'd, 'e) Form -> bool)
          (SMOStateInterp :state ->
                           (command inst, principal, 'd, 'e) Form)
          (certsr1a npriv privcmd (PR privcmd) :
             (command inst, principal, 'd, 'e) Form list) ins
          (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) <=>
     inputOKr1
       (Name (Staff Alice) quoting Name (Role Commander) says
        (prop (SOME (PR privcmd) :command inst) :
           (command inst, principal, 'd, 'e) Form)) /\
     CFGInterpret (M,Oi,Os)
       (CFG (inputOKr1 :(command inst, principal, 'd, 'e) Form -> bool)
          (SMOStateInterp :state ->
                           (command inst, principal, 'd, 'e) Form)
          (certsr1a npriv privcmd (PR privcmd) :
             (command inst, principal, 'd, 'e) Form list)
          (Name (Staff 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 ***
```

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.    *)
(* ======================================================================= *)


(***************************************************************************)
(* SM0_Commander_privcmd_trapped_lemma                                       *)
(***************************************************************************)
val SM0_Commander_privcmd_trapped_lemma =
TAC_PROOF(([],
``CFGInterpret ((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os)
  (CFG inputOK SM0StateInterp (certs npriv privcmd)
   (((Name (Role Commander)) says (prop (SOME (PR (privcmd:privcmd)))))::ins)
   s (outs:output list)) ==>
  ((M,Oi,Os) sat (prop NONE))``),
REWRITE_TAC[CFGInterpret_def,certs_def,SM0StateInterp_def,satList_CONS,
            satList_nil,sat_TT] THEN
PROVE_TAC[Controls, Modus_Ponens])

val _ = save_thm("SM0_Commander_privcmd_trapped_lemma",
SM0_Commander_privcmd_trapped_lemma)


(***************************************************************************)
(* SM0_Commander_trap_privcmd_justified_thm                                  *)
(***************************************************************************)
val SM0_Commander_trap_privcmd_justified_thm =
let
 val th1 =
 ISPECL
 [``inputOK:(command inst, principal,'d,'e)Form -> bool``,
  ``SM0StateInterp:state->(command inst, principal,'d,'e)Form``,
  ``(certs npriv privcmd):(command inst, principal,'d,'e)Form list``,
  ``Name (Role Commander)``,``PR privcmd``,
  ``ins:(command inst,principal,'d,'e)Form list``,
  ``s:state``,``outs:output list``]
 TR_trap_cmd_rule
in
 TAC_PROOF(([],
``!(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 (inputOK :(command inst , principal , 'd, 'e) Form -> bool)
        (SM0StateInterp :state ->
                         (command inst , principal , 'd, 'e) Form)
        (certs (npriv :npriv) privcmd :
           (command inst , principal , 'd, 'e) Form list)
        (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 (inputOK :(command inst , principal , 'd, 'e) Form -> bool)
        (SM0StateInterp :state ->
                         (command inst , principal , 'd, 'e) Form)
        (certs npriv privcmd :
           (command inst , principal , 'd, 'e) Form list) ins
        (NS s (trap (PR privcmd)))
        (Out s (trap (PR privcmd))::outs)) <=>
   inputOK
     (Name (Role Commander) says
      (prop (SOME (PR privcmd) :command inst) :
         (command inst , principal , 'd, 'e) Form)) /\
   CFGInterpret (M,Oi,Os)
     (CFG (inputOK :(command inst , principal , 'd, 'e) Form -> bool)
        (SM0StateInterp :state ->
                         (command inst , principal , 'd, 'e) Form)
        (certs npriv privcmd :
           (command inst , principal , 'd, 'e) Form list)
        (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)``),
 PROVE_TAC[th1,SM0_Commander_privcmd_trapped_lemma])
end

val _ = save_thm ("SM0_Commander_trap_privcmd_justified_thm",
SM0_Commander_trap_privcmd_justified_thm)

(****************************************************************************)
(* SM0r1_mapSM0_Alice_Commander_trap_privcmd_lemma                          *)
(****************************************************************************)
val th1=
TAC_PROOF(([],
``(((M:(command inst,'b,principal,'d,'e)Kripke),Oi,Os) satList
   (certsr1a npriv privcmd (PR privcmd))) ==>
  ((M,Oi,Os) sat ((Name (Staff Alice)) quoting (Name (Role Commander)) says
  (prop (SOME (PR (privcmd:privcmd))))))
  ==> ((M,Oi,Os) sat (prop NONE))``),
REWRITE_TAC
 [certsr1a_def,certs_def,satList_CONS,satList_nil,(GSYM satList_conj)] THEN
PROVE_TAC[Rep_Says, Modus_Ponens])

val SM0r1_mapSM0_Alice_Commander_trap_privcmd_lemma =
TAC_PROOF(([],
``CFGInterpret ((M:(command inst ,'b,principal ,'d,'e)Kripke) ,Oi,Os)
  (CFG inputOKr1 SM0StateInterp (certsr1a npriv privcmd (PR privcmd))
   (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] THEN
PROVE_TAC[th1])

val _ = save_thm("SM0r1_mapSM0_Alice_Commander_trap_privcmd_lemma",
SM0r1_mapSM0_Alice_Commander_trap_privcmd_lemma)


(**************************************************************************)
(* SM0r1_Commander_Alice_trap_privcmd_justified_thm                      *)
(**************************************************************************)
val SM0r1_Commander_Alice_trap_privcmd_justified_thm =
let
 val th1 =
 ISPECL
 [``inputOKr1:(command inst , principal ,'d,'e)Form -> bool``,
  ``SM0StateInterp:state ->(command inst , principal ,'d,'e)Form``,
  ``(certsr1a npriv privcmd (PR privcmd)):(command inst , principal ,'d,'e)Form list
      ``,
  ``(Name (Staff 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]
     SM0r1_mapSM0_Alice_Commander_trap_privcmd_lemma
in
 REWRITE_RULE[th2]th1
end

val _ =  save_thm  ("SM0r1_Commander_Alice_trap_privcmd_justified_thm",
 SM0r1_Commander_Alice_trap_privcmd_justified_thm)


(**************************************************************************)
(* SM0r1_Commander_mapSM0inputOperatorBob_trap_privcmd_justified_thm      *)
(**************************************************************************)
val SM0r1_Commander_mapSM0inputOperatorBob_trap_privcmd_justified_thm =
let
 val th1 =
  TAC_PROOF(([] ,
  ``(((Name (Staff Alice) quoting (Name (Role Commander)) says
    (prop (SOME (PR (privcmd:privcmd))))):(command inst , principal ,'d,'e)Form))
   =  mapSM0inputOperatorBob
    (((Name (Role Commander)) says (prop (SOME (PR (privcmd:privcmd))))):
    (command inst , principal ,'d,'e)Form)``),
   REWRITE_TAC[mapSM0inputOperatorBob_def])
in
 GEN_ALL(REWRITE_RULE[th1]SM0r1_Commander_Alice_trap_privcmd_justified_thm)
end

val _ =
save_thm
```

```
  ("SM0r1_Commander_mapSM0inputOperatorBob_trap_privcmd_justified_thm",
    SM0r1_Commander_mapSM0inputOperatorBob_trap_privcmd_justified_thm)

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

val _ = export_theory();



end (* structure *)
```