*Project:* DRA FRONT END FILTER PROJECT

*Title:* Proof of Security (I)

*Ref:* DS/FMU/FEF/009 *Issue: Revision* : *3.2* *Date:* 5 June 2016

*Status:* Approved *Type:* Specification

*Keywords:*

*Author:*

| *Name* | *Location* | *Signature* | *Date* |
|---|---|---|---|
| G. M. Prout | WIN01 | | |

*Authorisation for Issue:*

| *Name* | *Function* | *Signature* | *Date* |
|---|---|---|---|
| R.B. Jones | HAT Manager | | |

*Abstract:* This document contains the formal proof of the unwinding result, part of the proof of security of the SSQL Abstract Machine for the DRA front end filter project RSRE 1C/6130.

*Distribution*: HAT FEF File
Simon Wiseman

# 0 DOCUMENT CONTROL

## 0.1 Contents List



## 0.2 Document Cross References

[1] DS/FMU/017. *Secure Database Technical Proposal.* High Assurance Team, ICL Secure Systems, WIN01, 21st January 1992.

[2] DS/FMU/FEF/003. *Formal Security Policy.* G.M. Prout, ICL Secure Systems, WIN01.

[3] DS/FMU/FEF/005. *Specifications of hide and updateState.* G.M. Prout, ICL Secure Systems, WIN01.

[4] DS/FMU/FEF/006. *Security Conjecture for the SSQL Abstract Machine.* G.M. Prout, ICL Secure Systems, WIN01.

[5] DS/FMU/FEF/007. *Proof Strategy.* G.M. Prout, ICL Secure Systems, WIN01.

## 0.3 Changes History

**Issue** *Revision* : *3.2* **(5 June 2016)** Latest approved version.

**Issue 3.3** Removed dependency on ICL logo font

## 0.4 Changes Forecast

None.

# 1 GENERAL

## 1.1 Scope

This document provides a formal proof of the unwinding result as specified in the proof strategy [5]. It constitutes deliverable D5 of work package 1b, as given in section 7 of the Secure Database Technical Proposal, [1].

## 1.2 Introduction

This document is a proof script which provides a formal proof of the unwinding result described in the proof strategy document [5]. Proof of the unwinding result comprises proofs of the lemmas 3,4 and 5 from [5]:

**Lemma3**

$$\begin{array}{ll} ?\vdash & (hide \in secureHide \\ & \wedge \\ & (hide, updateState) \in secureUpdate) \\ & \Rightarrow \\ & SSQLtf \in secureStf \end{array}$$

**Lemma4**

$$\begin{array}{ll} ?\vdash & SSQLtf \in secureStf \\ & \Rightarrow \\ & (iterate\ SSQLtf) \in secureItf \end{array}$$

**Lemma5**

$$\begin{array}{ll} ?\vdash & (iterate\ SSQLtf) \in secureItf \\ & \Rightarrow \\ & behaviours\ SSQLam \in secure \end{array}$$

where *hide* and *updateState* are defined in [3], *secureHide*, *secureUpdate*, *SSQLtf*, *secureStf* and *secureItf* are defined in [5], *iterate*, *behaviours* and *SSQLam* are defined in [4], and *secure* is defined in [2].

# 2 PRELIMINARIES

The following ProofPower instructions set up the new theory *fef009*.

SML

```
open_theory "fef007";
(force_delete_theory "fef009" handle _ => ());
new_theory"fef009";
push_merge_pcs["hol","wrk049","'pair1"] ;
```

In the proof script that follows, some output from the proof tool has been included for clarity where top level goals are split into subgoals. The proof script proper is in Standard ML, and is marked by a vertical bar on the left starting with the characters "SML". Output from the proof tool is distinguished by a vertical bar on the left headed by "HOL output". The subgoals consist of a *conclusion* to be proven and a number of *assumptions* which are available for use in the proof. The assumptions are listed first, followed by the conclusion.

### 2.1 Retrieving the Constants

The definitions of constants used in the specifications are retrieved from their defining theories.

```
SML
val secureHide_def = get_spec⌜secureHide⌝;
val secureUpdate_def = conv_rule(MAP_C let_conv)(get_spec⌜secureUpdate⌝);
val SSQLtf_def = get_spec⌜SSQLtf⌝;
val secureStf_def = conv_rule(MAP_C let_conv)(get_spec⌜secureStf⌝);
val same_ins_def = conv_rule(MAP_C let_conv)(get_spec⌜same_ins⌝);
val same_outs_def = conv_rule(MAP_C let_conv)(get_spec⌜same_outs⌝);
val mkTf_def = get_spec⌜mkTf⌝;
val secureItf_def = conv_rule(MAP_C let_conv)(get_spec⌜secureItf⌝);
val behaviours_def = get_spec⌜behaviours⌝;
val SSQLam_def = get_spec⌜SSQLam⌝;
val secure_def = get_spec⌜secure⌝;
```

## 3 PROOF OF *Lemma3*

```
SML
push_goal([],⌜Lemma3⌝);
a(rewrite_tac[Lemma3,secureHide_def,secureUpdate_def,SSQLtf_def,secureStf_def,
        same_ins_def,same_outs_def]);
a(strip_tac THEN strip_tac);
```

HOL output

```
Tactic produced 2 subgoals:

(* *** Goal "1" *** *)

(*  7 *) ⌜∀ c1 c2 s1 s2
        • hide (c1, s1) = hide (c1, s2) ∧ c1 dominates c2
          ⇒ hide (c2, s1) = hide (c2, s2)⌝
(*  6 *) ⌜∀ c1 c2 s e
        • ¬ hide (c2, s) = hide (c2, Fst (updateState (c1, e, s)))
          ⇒ c2 dominates c1⌝
(*  5 *) ⌜∀ c1 c2 s1 s2 e
        • hide (c1, s1) = hide (c1, s2) ∧ c1 dominates c2
          ⇒ hide (c1, Fst (updateState (c2, e, s1)))
            = hide (c1, Fst (updateState (c2, e, s2)))⌝
(*  4 *) ⌜∀ c s1 s2 e
        • hide (c, s1) = hide (c, s2)
          ⇒ Snd (updateState (c, e, s1)) = Snd (updateState (c, e, s2))⌝
(*  3 *) ⌜∀ c s e• Fst (Snd (updateState (c, e, s))) = c⌝
(*  2 *) ⌜hide (c, s1) = hide (c, s2)⌝
(*  1 *) ⌜[i1] ↾ {(q, c')|c dominates c'}
          = [i2] ↾ {(q, c')|c dominates c'}⌝

(* ?⊢ *) ⌜∀ s1 s2 i1 i2 c
        • hide (c, s1) = hide (c, s2)
            ∧ [i1] ↾ {(q, c')|c dominates c'} = [i2] ↾ {(q, c')|c dominates c'}
          ⇒ hide(c, Fst(mkTf hide processQuery updateState(i1, s1)))
              = hide(c,Fst(mkTf hide processQuery updateState(i2, s2)))
            ∧ [Snd (mkTf hide processQuery updateState (i1, s1))]
                ↾ {(c', d)|c dominates c'}
              = [Snd (mkTf hide processQuery updateState (i2, s2))]
                ↾ {(c', d)|c dominates c'}⌝
```

SML

```
a(REPEAT ∀_tac THEN  strip_tac);
a(POP_ASM_T ante_tac THEN rewrite_tac[↾_thm6]);
a ⇒_tac;
```

HOL output

*Tactic produced 2 subgoals*:

(* *** *Goal "1.1"* *** *)

(* 7 *) ⌜∀ $c_1$ $c_2$ $s_1$ $s_2$
• *hide* ($c_1$, $s_1$) = *hide* ($c_1$, $s_2$) ∧ $c_1$ *dominates* $c_2$
⇒ *hide* ($c_2$, $s_1$) = *hide* ($c_2$, $s_2$)⌝
(* 6 *) ⌜∀ $c_1$ $c_2$ $s$ $e$
• ¬ *hide* ($c_2$, $s$) = *hide* ($c_2$, *Fst* (*updateState* ($c_1$, $e$, $s$)))
⇒ $c_2$ *dominates* $c_1$⌝
(* 5 *) ⌜∀ $c_1$ $c_2$ $s_1$ $s_2$ $e$
• *hide* ($c_1$, $s_1$) = *hide* ($c_1$, $s_2$) ∧ $c_1$ *dominates* $c_2$
⇒ *hide* ($c_1$, *Fst* (*updateState* ($c_2$, $e$, $s_1$)))
= *hide* ($c_1$, *Fst* (*updateState* ($c_2$, $e$, $s_2$)))⌝
(* 4 *) ⌜∀ $c$ $s_1$ $s_2$ $e$
• *hide* ($c$, $s_1$) = *hide* ($c$, $s_2$)
⇒ *Snd* (*updateState* ($c$, $e$, $s_1$)) = *Snd* (*updateState* ($c$, $e$, $s_2$))⌝
(* 3 *) ⌜∀ $c$ $s$ $e$• *Fst* (*Snd* (*updateState* ($c$, $e$, $s$))) = $c$⌝
(* 2 *) ⌜*hide* ($c$, $s_1$) = *hide* ($c$, $s_2$)⌝
(* 1 *) ⌜$i_1$ = $i_2$⌝

(* ?⊢ *) ⌜*hide* ($c$, *Fst* (*mkTf hide processQuery updateState* ($i_1$, $s_1$)))
= *hide*
($c$, *Fst* (*mkTf hide processQuery updateState* ($i_2$, $s_2$)))
∧ (*Snd* (*mkTf hide processQuery updateState* ($i_1$, $s_1$))
= *Snd* (*mkTf hide processQuery updateState* ($i_2$, $s_2$))
∨ ¬ $c$ *dominates Fst*
(*Snd* (*mkTf hide processQuery updateState* ($i_1$, $s_1$)))
∧ ¬ $c$ *dominates Fst* (*Snd*(*mkTf hide processQuery updateState*($i_2$, $s_2$))))⌝

SML

```
a(POP_ASM_T rewrite_thm_tac);
a(LEMMA_T`i2 = (Fst i2,Snd i2)` pure_once_rewrite_thm_tac
        THEN_LIST[rewrite_tac[],asm_rewrite_tac[mkTf_def]]);
a(cases_tac`c dominates Snd i2` THEN_TRY asm_rewrite_tac[]);
(* *** Goal "1.1.1" *** *)
a(lemma_tac`hide (Snd i2, s1) = hide (Snd i2, s2)`
        THEN_LIST[list_asm_ante_tac (map get_asm[1,2,7])THEN prove_tac[],
        TOP_ASM_T rewrite_thm_tac]);
a(list_asm_ante_tac (map get_asm[1,2,3,5,6])THEN prove_tac[]);
(* *** Goal "1.1.2" *** *)
a(list_spec_nth_asm_tac 6[`Snd i2`,`c`,`s1`,
        `processQuery (Fst i2, Snd i2, hide (Snd i2, s1))`]);
a(POP_ASM_T (rewrite_thm_tac o eq_sym_rule));
a(list_spec_nth_asm_tac 6[`Snd i2:Class`,`c`,`s2`,
        `processQuery (Fst i2, Snd i2, hide (Snd i2, s2))`]);
a(POP_ASM_T (rewrite_thm_tac o eq_sym_rule));
a(DROP_NTH_ASM_T 2 rewrite_thm_tac);
```

HOL output

```
(* *** Goal "1.2" *** *)

(*  8 *) ⌜∀ c1 c2 s1 s2
         • hide (c1, s1) = hide (c1, s2) ∧ c1 dominates c2
           ⇒ hide (c2, s1) = hide (c2, s2)⌝
(*  7 *) ⌜∀ c1 c2 s e
         • ¬ hide (c2, s) = hide (c2, Fst (updateState (c1, e, s)))
           ⇒ c2 dominates c1⌝
(*  6 *) ⌜∀ c1 c2 s1 s2 e
         • hide (c1, s1) = hide (c1, s2) ∧ c1 dominates c2
           ⇒ hide (c1, Fst (updateState (c2, e, s1)))
             = hide (c1, Fst (updateState (c2, e, s2)))⌝
(*  5 *) ⌜∀ c s1 s2 e
         • hide (c, s1) = hide (c, s2)
           ⇒ Snd (updateState (c, e, s1)) = Snd (updateState (c, e, s2))⌝
(*  4 *) ⌜∀ c s e• Fst (Snd (updateState (c, e, s))) = c⌝
(*  3 *) ⌜hide (c, s1) = hide (c, s2)⌝
(*  2 *) ⌜¬ c dominates Snd i1⌝
(*  1 *) ⌜¬ c dominates Snd i2⌝

(* ?⊢ *) ⌜hide (c, Fst (mkTf hide processQuery updateState (i1, s1)))
           = hide
             (c, Fst (mkTf hide processQuery updateState (i2, s2)))
         ∧ (Snd (mkTf hide processQuery updateState (i1, s1))
           = Snd (mkTf hide processQuery updateState (i2, s2))
         ∨ ¬ c dominates Fst
                (Snd (mkTf hide processQuery updateState (i1, s1)))
           ∧ ¬ c dominates Fst(Snd(mkTf hide processQuery updateState(i2, s2))))⌝
```

SML

```
a(LEMMA_T⌜i1 = (Fst i1,Snd i1)⌝ pure_once_rewrite_thm_tac
        THEN_LIST[rewrite_tac[],id_tac]);
a(LEMMA_T⌜i2 = (Fst i2,Snd i2)⌝ pure_once_rewrite_thm_tac
        THEN_LIST[rewrite_tac[],asm_rewrite_tac[mkTf_def]]);
a(list_spec_nth_asm_tac 7[⌜Snd i1⌝,⌜c⌝,⌜s1⌝,
        ⌜processQuery (Fst i1, Snd i1, hide (Snd i1, s1))⌝]);
a(POP_ASM_T (rewrite_thm_tac o eq_sym_rule));
a(list_spec_nth_asm_tac 7[⌜Snd i2⌝,⌜c⌝,⌜s2⌝,
        ⌜processQuery (Fst i2, Snd i2, hide (Snd i2, s2))⌝]);
a(POP_ASM_T (rewrite_thm_tac o eq_sym_rule));
a(DROP_NTH_ASM_T 3 rewrite_thm_tac);
```

HOL output

```
(* *** Goal "2" *** *)

(*  5 *)  ⌜∀ c1 c2 s1 s2
          • hide (c1, s1) = hide (c1, s2) ∧ c1 dominates c2
             ⇒ hide (c2, s1) = hide (c2, s2)⌝
(*  4 *)  ⌜∀ c1 c2 s e
          • ¬ hide (c2, s) = hide (c2, Fst (updateState (c1, e, s)))
             ⇒ c2 dominates c1⌝
(*  3 *)  ⌜∀ c1 c2 s1 s2 e
          • hide (c1, s1) = hide (c1, s2) ∧ c1 dominates c2
             ⇒ hide (c1, Fst (updateState (c2, e, s1)))
               = hide (c1, Fst (updateState (c2, e, s2)))⌝
(*  2 *)  ⌜∀ c s1 s2 e
          • hide (c, s1) = hide (c, s2)
             ⇒ Snd (updateState (c, e, s1)) = Snd (updateState (c, e, s2))⌝
(*  1 *)  ⌜∀ c s e• Fst (Snd (updateState (c, e, s))) = c⌝

(* ?⊢ *)  ⌜∀ s i c • ¬ c dominates Snd i
             ⇒ hide (c, s) = hide(c, Fst (mkTf hide processQuery updateState (i, s)))
               ∧ ¬ c dominates Fst(Snd (mkTf hide processQuery updateState (i, s)))⌝
```

SML

```
a(REPEAT ∀_tac THEN  strip_tac);
a(LEMMA_T⌜i = (Fst i,Snd i)⌝ pure_once_rewrite_thm_tac
        THEN_LIST[rewrite_tac[],asm_rewrite_tac[mkTf_def]]);
a(list_asm_ante_tac (map get_asm[1,5]) THEN prove_tac[]);
val lemma3_thm = save_pop_thm"Lemma3";
```

# 4 PROOF OF *Lemma4*

## 4.1 Auxiliary Results

We first prove the consistency of *iterate* by providing a witness *iterate_witness*, for use in the consistency proof. This definition is a primitive recursive definition whose consistency is automatically proven by the system.

HOL Constant

```
iterate_witness : ((('QUERY × Class) × 'STATE) → ('STATE × (Class × 'DATA)))
                        → 'STATE → ('QUERY × Class)LIST
                        → ('STATE × ((Class × 'DATA)LIST))
```

---

```
∀s:'STATE;i:'QUERY × Class;si:('QUERY × Class) LIST;
f:(('QUERY × Class) × 'STATE) → ('STATE × (Class × 'DATA))
•       iterate_witness f s [] = (s,[])
∧
        iterate_witness f s (Cons i si)
                =
                (Fst(f(i,Fst(iterate_witness f s si))),
                (Snd(iterate_witness f s si) ⌢
                        [Snd(f(i,Fst(iterate_witness f s si)))] ))
```

SML

```
push_consistency_goal⌜iterate⌝;
a(conv_tac(MAP_C let_conv));
a(∃_tac⌜λx (y,z) • iterate_witness x z (Rev y)⌝);
a(rewrite_tac[⌢_Cons_thm,rev_def,rev_sym_thm,get_spec⌜iterate_witness⌝]);
val iterate_consistent = save_consistency_thm⌜iterate⌝ (pop_thm());
```

HOL output

```
iterate_consistent = ⊢ Consistent
    (λ iterate'
     • ∀ s i si f
       • iterate' f ([], s) = (s, [])
          ∧ (let sso = iterate' f (si, s)
          in iterate' f (si ⌢ [i], s)
            = (Fst (f (i, Fst sso)), Snd sso ⌢ [Snd (f (i, Fst sso))]))) : THM
```

Now the definition of *iterate* is retrieved with its consistency proof obligation discharged.

SML

```
val iterate_def = conv_rule(MAP_C let_conv)(get_spec⌜iterate⌝);
```

HOL output

```
iterate_def = ⊢ ∀ s i si f
  • iterate f ([], s) = (s, [])
      ∧ iterate f (si ⌢ [i], s)
        = (Fst (f (i, Fst (iterate f (si, s)))),
           Snd (iterate f (si, s)) ⌢ [Snd (f (i, Fst (iterate f (si, s))))])
```

## 4.2 Lemma4

SML

```
push_goal([],⌜Lemma4⌝);
a(rewrite_tac[Lemma4,secureStf_def,secureItf_def]);
a strip_tac;
(** need to reorder variables **)
a(REPEAT ∀_tac);
a(MAP_EVERY intro_∀_tac[(⌜s2⌝,⌜s2⌝),(⌜s1⌝,⌜s1⌝),(⌜c⌝,⌜c⌝),(⌜si2⌝,⌜si2⌝)]);
a(REV_LIST_INDUCTION_T⌜si1⌝asm_tac);
```

HOL output

```
Tactic produced 2 subgoals:

(* *** Goal "1" *** *)

(*  2 *)  ⌜∀ s1 s2 i1 i2 c
          • hide (c, s1) = hide (c, s2) ∧ ([i1], [i2]) ∈ same_ins c
             ⇒ hide (c, Fst (SSQLtf (i1, s1)))
                = hide (c, Fst (SSQLtf (i2, s2)))
               ∧ ([Snd (SSQLtf (i1, s1))], [Snd (SSQLtf (i2, s2))]) ∈ same_outs c⌝
(*  1 *)  ⌜∀ s i c
          • ¬ c dominates Snd i
             ⇒ hide (c, s) = hide (c, Fst (SSQLtf (i, s)))
               ∧ ¬ c dominates Fst (Snd (SSQLtf (i, s)))⌝

(* ?⊢ *)  ⌜∀ si2 c s1 s2
          • hide (c, s1) = hide (c, s2) ∧ ([], si2) ∈ same_ins c
             ⇒ hide (c, Fst (iterate SSQLtf ([], s1)))
                = hide (c, Fst (iterate SSQLtf (si2, s2)))
               ∧ (Snd (iterate SSQLtf ([], s1)),
                  Snd (iterate SSQLtf (si2, s2))) ∈ same_outs c⌝
```

SML

```
a ∀_tac;
a(REV_LIST_INDUCTION_T⌜si2⌝asm_tac);
(* *** Goal "1.1" *** *)
(* *** si1, si2 both empty *** *)
a(rewrite_tac[iterate_def,same_ins_def,same_outs_def]);
(* *** Goal "1.2" *** *)
(* *** si1 empty, si2 non empty *** *)
a(REPEAT ∀_tac);
a(POP_ASM_T ante_tac);
a(rewrite_tac[iterate_def,same_ins_def,same_outs_def]);
a(cases_tac⌜c dominates Snd last⌝THEN asm_rewrite_tac[]);
a(lemma_tac⌜(hide (c, Fst (iterate SSQLtf (si2, s2)))
            = hide (c, Fst (SSQLtf (last, Fst (iterate SSQLtf (si2, s2))))))
        ∧ ¬(c dominates Fst
                        (Snd (SSQLtf (last, Fst (iterate SSQLtf (si2, s2))))))⌝
        THEN_LIST[asm_prove_tac[],asm_rewrite_tac[]]);
a ⇒_tac;
a(⇒_tac THEN asm_rewrite_tac[]);
a(DROP_NTH_ASM_T 3 (ante_tac o list_∀_elim[⌜c⌝,⌜s1⌝,⌜s2⌝]) THEN asm_rewrite_tac[]);
```

HOL output

```
(* *** Goal "2" *** *)

(*  3 *)  ⌜∀ s1 s2 i1 i2 c
          • hide (c, s1) = hide (c, s2) ∧ ([i1], [i2]) ∈ same_ins c
            ⇒ hide (c, Fst (SSQLtf (i1, s1)))
                = hide (c, Fst (SSQLtf (i2, s2)))
              ∧ ([Snd (SSQLtf (i1, s1))], [Snd (SSQLtf (i2, s2))])
                ∈ same_outs c⌝
(*  2 *)  ⌜∀ s i c
          • ¬ c dominates Snd i
            ⇒ hide (c, s) = hide (c, Fst (SSQLtf (i, s)))
              ∧ ¬ c dominates Fst (Snd (SSQLtf (i, s)))⌝
(*  1 *)  ⌜∀ si2 c s1 s2
          • hide (c, s1) = hide (c, s2) ∧ (si1, si2) ∈ same_ins c
            ⇒ hide (c, Fst (iterate SSQLtf (si1, s1)))
                = hide (c, Fst (iterate SSQLtf (si2, s2)))
              ∧ (Snd (iterate SSQLtf (si1, s1)),
                 Snd (iterate SSQLtf (si2, s2))) ∈ same_outs c⌝

(* ?⊢ *)  ⌜∀ last si2 c s1 s2
          • hide (c, s1) = hide (c, s2) ∧ (si1 ⌢ [last], si2) ∈ same_ins c
            ⇒ hide (c, Fst (iterate SSQLtf (si1 ⌢ [last], s1)))
                = hide (c, Fst (iterate SSQLtf (si2, s2)))
              ∧ (Snd (iterate SSQLtf (si1 ⌢ [last], s1)),
                 Snd (iterate SSQLtf (si2, s2))) ∈ same_outs c⌝
```

SML

```
(** need to reorder variables **)
a(REPEAT ∀_tac);
a(MAP_EVERY intro_∀_tac[(⌜s2⌝,⌜s2⌝),(⌜s1⌝,⌜s1⌝),(⌜c⌝,⌜c⌝),(⌜last⌝,⌜last⌝)]);
a(POP_ASM_T ante_tac);
a(REV_LIST_INDUCTION_T⌜si2⌝asm_tac);
(* *** Goal "2.1" *** *)
(* *** si1 non empty, si2 empty *** *)
a(rewrite_tac[iterate_def,same_ins_def,same_outs_def]);
a(⇒_tac THEN REPEAT ∀_tac);
a(cases_tac⌜c dominates Snd last⌝THEN asm_rewrite_tac[]);
a(lemma_tac⌜(hide (c, Fst (iterate SSQLtf (si1, s1)))
            = hide (c, Fst (SSQLtf (last, Fst (iterate SSQLtf (si1, s1))))))
        ∧ ¬(c dominates Fst
                    (Snd (SSQLtf (last, Fst (iterate SSQLtf (si1, s1))))))⌝
        THEN_LIST[list_asm_ante_tac (map get_asm[1,3]) THEN prove_tac[],
              asm_rewrite_tac[]]);
a ⇒_tac;
a(DROP_NTH_ASM_T 6 (ante_tac o list_∀_elim[⌜[]:(Query × Class)LIST⌝,⌜c⌝,⌜s1⌝,⌜s2⌝])
        THEN asm_rewrite_tac[iterate_def]);
```

SML

```
(* *** Goal "2.2" *** *)
(* *** si1, si2 both non empty *** *)
a (∀_tac THEN POP_ASM_T ante_tac);
a(rewrite_tac[iterate_def,same_ins_def,same_outs_def]);
a(⇒_T asm_tac THEN ⇒_tac);
a(REPEAT ∀_tac);
a(cases_tac⌜c dominates Snd last⌝THEN cases_tac⌜c dominates Snd last'⌝
        THEN asm_rewrite_tac[] THEN ⇒_tac);
```

HOL output

*Tactic produced 4 subgoals*:

(* *** *Goal "2.2.1"* *** *)

(* 9 *) $\ulcorner \forall\ s_1\ s_2\ i_1\ i_2\ c$
$\bullet\ hide\ (c,\ s_1) = hide\ (c,\ s_2) \wedge ([i_1],\ [i_2]) \in same\_ins\ c$
$\Rightarrow hide\ (c,\ Fst\ (SSQLtf\ (i_1,\ s_1))) = hide\ (c,\ Fst\ (SSQLtf\ (i_2,\ s_2)))$
$\wedge ([Snd\ (SSQLtf\ (i_1,\ s_1))],\ [Snd\ (SSQLtf\ (i_2,\ s_2))]) \in same\_outs\ c \urcorner$

(* 8 *) $\ulcorner \forall\ s\ i\ c \bullet \neg\ c\ dominates\ Snd\ i$
$\Rightarrow hide\ (c,\ s) = hide\ (c,\ Fst\ (SSQLtf\ (i,\ s)))$
$\wedge \neg\ c\ dominates\ Fst\ (Snd\ (SSQLtf\ (i,\ s))) \urcorner$

(* 7 *) $\ulcorner (\forall\ si_2\ c\ s_1\ s_2 \bullet hide\ (c,\ s_1) = hide\ (c,\ s_2)$
$\wedge\ si_1 \upharpoonright \{(q,\ c') | c\ dominates\ c'\}$
$= si_2 \upharpoonright \{(q,\ c') | c\ dominates\ c'\}$
$\Rightarrow hide\ (c,\ Fst\ (iterate\ SSQLtf\ (si_1,\ s_1)))$
$= hide\ (c,\ Fst\ (iterate\ SSQLtf\ (si_2,\ s_2)))$
$\wedge\ Snd\ (iterate\ SSQLtf\ (si_1,\ s_1)) \upharpoonright \{(c',\ d) | c\ dominates\ c'\}$
$= Snd\ (iterate\ SSQLtf\ (si_2,\ s_2)) \upharpoonright \{(c',\ d) | c\ dominates\ c'\})$
$\Rightarrow (\forall\ last\ c\ s_1\ s_2 \bullet hide\ (c,\ s_1) = hide\ (c,\ s_2)$
$\wedge\ (if\ c\ dominates\ Snd\ last$
$then\ (si_1 \upharpoonright \{(q,\ c') | c\ dominates\ c'\}) \frown [last]$
$else\ si_1 \upharpoonright \{(q,\ c') | c\ dominates\ c'\})$
$= si_2 \upharpoonright \{(q,\ c') | c\ dominates\ c'\}$
$\Rightarrow hide(c, Fst(SSQLtf(last, Fst\ (iterate\ SSQLtf\ (si_1,\ s_1)))))$
$= hide\ (c,\ Fst\ (iterate\ SSQLtf\ (si_2,\ s_2)))$
$\wedge\ (if\ c\ dominates\ Fst(Snd(SSQLtf(last,$
$Fst\ (iterate\ SSQLtf\ (si_1,\ s_1)))))$
$then$
$(Snd\ (iterate\ SSQLtf\ (si_1,\ s_1)) \upharpoonright \{(c',\ d) | c\ dominates\ c'\})$
$\frown [Snd(SSQLtf(last, Fst\ (iterate\ SSQLtf\ (si_1,\ s_1))))]$
$else$
$Snd\ (iterate\ SSQLtf\ (si_1,\ s_1)) \upharpoonright \{(c',\ d) | c\ dominates\ c'\})$
$= Snd\ (iterate\ SSQLtf\ (si_2,\ s_2)) \upharpoonright \{(c',\ d) | c\ dominates\ c'\}) \urcorner$

(* 6 *) $\ulcorner \forall\ si_2\ c\ s_1\ s_2 \bullet hide\ (c,\ s_1) = hide\ (c,\ s_2)$
$\wedge\ si_1 \upharpoonright \{(q,\ c') | c\ dominates\ c'\} = si_2 \upharpoonright \{(q,\ c') | c\ dominates\ c'\}$
$\Rightarrow hide\ (c,\ Fst\ (iterate\ SSQLtf\ (si_1,\ s_1)))$
$= hide\ (c,\ Fst\ (iterate\ SSQLtf\ (si_2,\ s_2)))$
$\wedge\ Snd\ (iterate\ SSQLtf\ (si_1,\ s_1)) \upharpoonright \{(c',\ d) | c\ dominates\ c'\}$
$= Snd\ (iterate\ SSQLtf\ (si_2,\ s_2)) \upharpoonright \{(c',\ d) | c\ dominates\ c'\} \urcorner$

(* 5 *) $\ulcorner c\ dominates\ Snd\ last \urcorner$

(* 4 *) $\ulcorner c\ dominates\ Snd\ last' \urcorner$

(* 3 *) ~~$\ulcorner hide\ (c,\ s_1) = hide\ (c,\ s_2) \urcorner$~~

(* 2 *) $\ulcorner si_1 \upharpoonright \{(q,\ c') | c\ dominates\ c'\} = si_2 \upharpoonright \{(q,\ c') | c\ dominates\ c'\} \urcorner$

(* 1 *) $\ulcorner last' = last \urcorner$

HOL output

```
(* ?⊢ *)  ⌜hide(c,Fst(SSQLtf(last′, Fst (iterate SSQLtf (si1, s1)))))
          = hide(c,Fst(SSQLtf(last, Fst (iterate SSQLtf (si2, s2)))))
           ∧ (if c dominates Fst(Snd(SSQLtf(last′,Fst(iterate SSQLtf (si1, s1)))))
              then
                (Snd (iterate SSQLtf (si1, s1)) ↾ {(c′, d)|c dominates c′})
                  ⌢ [Snd(SSQLtf
                          (last′, Fst (iterate SSQLtf (si1, s1))))]
              else
                Snd (iterate SSQLtf (si1, s1)) ↾ {(c′, d)|c dominates c′})
             = (if c dominates Fst(Snd(SSQLtf(last, Fst (iterate SSQLtf (si2, s2)))))
              then
                (Snd (iterate SSQLtf (si2, s2)) ↾ {(c′, d)|c dominates c′})
                  ⌢ [Snd(SSQLtf(last, Fst (iterate SSQLtf (si2, s2))))]
              else
                Snd (iterate SSQLtf (si2, s2)) ↾ {(c′, d)|c dominates c′})⌝
```

SML

```
a(POP_ASM_T rewrite_thm_tac);
a(DROP_NTH_ASM_T 5(ante_tac o list_∀_elim[⌜si2⌝,⌜c⌝,⌜s1⌝,⌜s2⌝]));
a(asm_rewrite_tac[] THEN ⇒_tac);
a(DROP_NTH_ASM_T 9(asm_tac o rewrite_rule[same_ins_def,same_outs_def]));
a(list_spec_nth_asm_tac 1 [⌜Fst (iterate SSQLtf (si1, s1))⌝,
      ⌜Fst (iterate SSQLtf (si2, s2))⌝,⌜last⌝,⌜last⌝,⌜c⌝]);
a(POP_ASM_T (ante_tac o rewrite_rule[↾_thm6]));
a(REPEAT strip_tac THEN asm_rewrite_tac[]);
```

HOL output

```
(* *** Goal "2.2.2" *** *)

(*  8 *) ⌜∀ s1 s2 i1 i2 c
        • hide (c, s1) = hide (c, s2) ∧ ([i1], [i2]) ∈ same_ins c
          ⇒ hide (c, Fst (SSQLtf (i1, s1))) = hide (c, Fst (SSQLtf (i2, s2)))
            ∧ ([Snd (SSQLtf (i1, s1))], [Snd (SSQLtf (i2, s2))]) ∈ same_outs c⌝
.
.
.
(*  4 *) ⌜c dominates Snd last⌝
(*  3 *) ⌜¬ c dominates Snd last'⌝
(*  2 *) ⌜hide (c, s1) = hide (c, s2)⌝
(*  1 *) ⌜si1 ↾ {(q, c')|c dominates c'}
          = (si2 ↾ {(q, c')|c dominates c'}) ⌢ [last]⌝


(* ?⊢ *) ⌜hide(c,Fst(SSQLtf(last', Fst (iterate SSQLtf (si1, s1)))))
          = hide(c,Fst(SSQLtf(last, Fst (iterate SSQLtf (si2, s2)))))
           ∧ (if c dominates Fst(Snd(SSQLtf(last',Fst(iterate SSQLtf (si1, s1)))))
              then
                (Snd (iterate SSQLtf (si1, s1)) ↾ {(c', d)|c dominates c'})
                  ⌢ [Snd(SSQLtf
                          (last', Fst (iterate SSQLtf (si1, s1))))]
              else
                Snd (iterate SSQLtf (si1, s1)) ↾ {(c', d)|c dominates c'})
            = (if c dominates Fst(Snd(SSQLtf(last, Fst (iterate SSQLtf (si2, s2)))))
              then
                (Snd (iterate SSQLtf (si2, s2)) ↾ {(c', d)|c dominates c'})
                  ⌢ [Snd(SSQLtf(last, Fst (iterate SSQLtf (si2, s2))))]
              else
                Snd (iterate SSQLtf (si2, s2)) ↾ {(c', d)|c dominates c'})⌝
```

SML

```
a(lemma_tac⌜(hide (c, Fst (iterate SSQLtf (si1, s1)))
        = hide (c, Fst (SSQLtf (last', Fst (iterate SSQLtf (si1, s1))))))
        ∧ ¬(c dominates Fst
                    (Snd (SSQLtf (last', Fst (iterate SSQLtf (si1, s1))))))⌝
        THEN_LIST[list_asm_ante_tac (map get_asm[3,7]) THEN prove_tac[],
              asm_rewrite_tac[]]);
a(DROP_NTH_ASM_T 7 (ante_tac o list_∀_elim[⌜si2 ⌢ [last]⌝,⌜c⌝,⌜s1⌝,⌜s2⌝]));
a(asm_rewrite_tac[iterate_def]);
```

HOL output

```
(* *** Goal "2.2.3" *** *)

(*  8 *) ⌜∀ s1 s2 i1 i2 c
         • hide (c, s1) = hide (c, s2) ∧ ([i1], [i2]) ∈ same_ins c
           ⇒ hide (c, Fst (SSQLtf (i1, s1))) = hide (c, Fst (SSQLtf (i2, s2)))
             ∧ ([Snd (SSQLtf (i1, s1))], [Snd (SSQLtf (i2, s2))]) ∈ same_outs c⌝
.
.
.
(*  4 *) ⌜¬ c dominates Snd last⌝
(*  3 *) ⌜c dominates Snd last'⌝
(*  2 *) ⌜hide (c, s1) = hide (c, s2)⌝
(*  1 *) ⌜(si1 ↾ {(q, c')|c dominates c'}) ⌢ [last'] = si2 ↾ {(q, c')|c dominates c'}⌝


(* ?⊢ *) ⌜hide(c,Fst(SSQLtf(last', Fst (iterate SSQLtf (si1, s1)))))
          = hide(c,Fst(SSQLtf(last, Fst (iterate SSQLtf (si2, s2)))))
           ∧ (if c dominates Fst(Snd(SSQLtf(last',Fst(iterate SSQLtf (si1, s1)))))
              then
                (Snd (iterate SSQLtf (si1, s1)) ↾ {(c', d)|c dominates c'})
                  ⌢ [Snd(SSQLtf
                        (last', Fst (iterate SSQLtf (si1, s1))))]
              else
                Snd (iterate SSQLtf (si1, s1)) ↾ {(c', d)|c dominates c'})
             = (if c dominates Fst(Snd(SSQLtf(last, Fst (iterate SSQLtf (si2, s2)))))
              then
                (Snd (iterate SSQLtf (si2, s2)) ↾ {(c', d)|c dominates c'})
                  ⌢ [Snd(SSQLtf(last, Fst (iterate SSQLtf (si2, s2))))]
              else
                Snd (iterate SSQLtf (si2, s2)) ↾ {(c', d)|c dominates c'})⌝
```

SML

```
a(lemma_tac⌜(hide (c, Fst (iterate SSQLtf (si₂, s₂)))
        = hide (c, Fst (SSQLtf (last, Fst (iterate SSQLtf (si₂, s₂))))))
        ∧ ¬(c dominates Fst
                    (Snd (SSQLtf (last, Fst (iterate SSQLtf (si₂, s₂))))))⌝
        THEN_LIST[list_asm_ante_tac (map get_asm[4,7]) THEN prove_tac[],
              asm_rewrite_tac[]]);
a(DROP_NTH_ASM_T 8 ante_tac THEN DROP_NTH_ASM_T 7 rewrite_thm_tac);
a(⇒_T (ante_tac o list_∀_elim[⌜last'⌝,⌜c⌝,⌜s₁⌝,⌜s₂⌝]));
a(asm_rewrite_tac[]);
```

HOL output

```
(* *** Goal "2.2.4" *** *)

(*  8 *)  ⌜∀ s₁ s₂ i₁ i₂ c
          • hide (c, s₁) = hide (c, s₂) ∧ ([i₁], [i₂]) ∈ same_ins c
            ⇒ hide (c, Fst (SSQLtf (i₁, s₁))) = hide (c, Fst (SSQLtf (i₂, s₂)))
              ∧ ([Snd (SSQLtf (i₁, s₁))], [Snd (SSQLtf (i₂, s₂))]) ∈ same_outs c⌝
.
.
.
(*  4 *)  ⌜¬ c dominates Snd last⌝
(*  3 *)  ⌜¬ c dominates Snd last'⌝
(*  2 *)  ⌜hide (c, s₁) = hide (c, s₂)⌝
(*  1 *)  ⌜si₁ ↾ {(q, c')|c dominates c'} = si₂ ↾ {(q, c')|c dominates c'}⌝


(* ?⊢ *)  ⌜hide(c,Fst(SSQLtf(last', Fst (iterate SSQLtf (si₁, s₁)))))
           = hide(c,Fst(SSQLtf(last, Fst (iterate SSQLtf (si₂, s₂)))))
            ∧ (if c dominates Fst(Snd(SSQLtf(last',Fst(iterate SSQLtf (si₁, s₁)))))
               then (Snd (iterate SSQLtf (si₁, s₁)) ↾ {(c', d)|c dominates c'})
                  ⌢ [Snd(SSQLtf(last', Fst (iterate SSQLtf (si₁, s₁))))]
               else  Snd (iterate SSQLtf (si₁, s₁)) ↾ {(c', d)|c dominates c'})
              = (if c dominates Fst(Snd(SSQLtf(last, Fst (iterate SSQLtf (si₂, s₂)))))
               then (Snd (iterate SSQLtf (si₂, s₂)) ↾ {(c', d)|c dominates c'})
                  ⌢ [Snd(SSQLtf(last, Fst (iterate SSQLtf (si₂, s₂))))]
               else Snd (iterate SSQLtf (si₂, s₂)) ↾ {(c', d)|c dominates c'})⌝
```

SML

```
a(lemma_tac ⌜(hide (c, Fst (iterate SSQLtf (si1, s1)))
        = hide (c, Fst (SSQLtf (last', Fst (iterate SSQLtf (si1, s1))))))
        ∧ ¬(c dominates Fst
                    (Snd (SSQLtf (last', Fst (iterate SSQLtf (si1, s1))))))
        ∧ (hide (c, Fst (iterate SSQLtf (si2, s2)))
        = hide (c, Fst (SSQLtf (last, Fst (iterate SSQLtf (si2, s2))))))
        ∧ ¬(c dominates Fst
                    (Snd (SSQLtf (last, Fst (iterate SSQLtf (si2, s2))))))⌝
        THEN_LIST[list_asm_ante_tac (map get_asm[3,4,7]) THEN prove_tac[],
              asm_rewrite_tac[]]);
a(DROP_NTH_ASM_T 2 (rewrite_thm_tac o eq_sym_rule));
a(DROP_NTH_ASM_T 3 (rewrite_thm_tac o eq_sym_rule));
a(list_asm_ante_tac (map get_asm[3,4,7]) THEN prove_tac[]);
val lemma4_thm = save_pop_thm"Lemma4";
```

# 5 PROOF OF *Lemma5*

SML

```
push_goal([],⌜Lemma5⌝);
a(rewrite_tac[Lemma5,secureItf_def,behaviours_def,SSQLam_def,
        secure_def,eq_sym_rule SSQLtf_def]);
a(REPEAT strip_tac);
a(list_spec_nth_asm_tac 2 [⌜isstate⌝,⌜isstate⌝,⌜si1⌝,⌜si2⌝,⌜clear⌝]);
val lemma5_thm = save_pop_thm"Lemma5";
```

# 6 REMAINING PROOF

Taking the main theorem to be proven, from the proof strategy document,[5], we can simplify the result using the proofs of *Lemma3*, *Lemma4* and *Lemma5*.

SML

```
val main_thm1 = save_thm("main_thm1",
        rewrite_rule[lemma3_thm,lemma4_thm,lemma5_thm]main_thm);
```

HOL output

```
main_thm1 = ⊢  Lemma1 ⇒ behaviours SSQLam ∈ secure
```

where *Lemma1*, from the proof strategy document [5], is:

```
        ?⊢      hide ∈ secureHide ∧ (hide,updateState) ∈ secureUpdate
```

# 7 CLOSING DOWN

The following ProofPower instruction restores the previous proof context.

```
SML
|pop_pc();
```

# 8 THE THEORY fef009

## 8.1 Parents

*fef007*

## 8.2 Children

*fef010*

## 8.3 Constants

**iterate_witness**

$$((' QUERY \times Class) \times ' STATE \to ' STATE \times Class \times ' DATA) \to ' STATE \to (' QUERY \times Class)\ LIST \to ' STATE \times (Class \times ' DATA)\ LIST$$

## 8.4 Definitions

**iterate_witness**

$$\vdash \forall\ s\ i\ si\ f \bullet iterate\_witness\ f\ s\ [] = (s, []) \wedge iterate\_witness\ f\ s\ (Cons\ i\ si) = (Fst\ (f\ (i, Fst\ (iterate\_witness\ f\ s\ si))), Snd\ (iterate\_witness\ f\ s\ si) @ [Snd\ (f\ (i, Fst\ (iterate\_witness\ f\ s\ si)))])$$

## 8.5 Theorems

**Lemma3** $\vdash Lemma3$

**iterate_consistent**

$$\vdash Consistent\ (\lambda\ iterate' \bullet \forall\ s\ i\ si\ f \bullet iterate'\ f\ ([], s) = (s, []) \wedge (let\ sso = iterate'\ f\ (si, s)\ in\ iterate'\ f\ (si @ [i], s) = (Fst\ (f\ (i, Fst\ sso)), Snd\ sso @ [Snd\ (f\ (i, Fst\ sso))])))$$

**Lemma4** $\vdash$ *Lemma4*
**Lemma5** $\vdash$ *Lemma5*
**main_thm1** $\vdash$ *Lemma1* $\Rightarrow$ *behaviours SSQLam* $\in$ *secure*

# 9 INDEX