Sample Examination Questions for MPRI 2-30

(All documents are allowed; duration: 3h)

1 Non-Interference

We consider the label lattice $\{L, H\}$ with $L \leq H$. For each of the following program, indicate whether it satisfies non-interference by either building a typing derivation using the rules in Appendix A or showing where typing fails. Additionally, explain whether the program satisfies the constant-time programming discipline.

1. $x : H\ var, y : H\ var \vdash$ **if** $x = 1$ **then** $y := 1$ **else** $y := 0$
2. $x : H\ var, y : L\ var \vdash$ **if** $x = 5$ **then** $x := 3$ **else** $y := 0$
3. $x : H\ var, y : L\ var \vdash$ **while** $x < 10$ **do** (**if** $y = 2$ **then** $x := x+1$ **else** $x := x + 2$)
4. $x : L\ var, y : H\ var \vdash$ **while** $x < 10$ **do** $(x := x + 1; y := y + 1)$

For each of the following programs, explain whether it satisfies speculative constant-time, or why it might be susceptible to speculative side-channel attacks. You can use the rules in Appendix B to justify your answer, however, building complete typing derivations is not required. All variables except s and *sec* will be considered as public, and we assume that the mask ms is well-initialized.

1.
```
if i < 5 {
  s[i] = sec;
}
x = p[0];
w[x] = 0;
```

2.
```
if i < 10 {
  ms = set_ms(i < 10);
  x = p[i];
  x = protect(x, ms);
}
w[x] = 0;
```

2 Label-based Verification

Question 1. Consider the following signature for (asymmetric) encryption

```
val enc (l1 l2: label) (msg: bytes l1) (k: pub_key l2) : bytes Public
```

What relation do we need on labels l_1 and l_2 to model a "secure" encryption?

Question 2. In your own words, explain the labels in the (simplified) signature of Diffie-Hellman shown below

```
val dh (l1: label) (priv: dh_private_key l1) (l2: label) (pub: dh_public_key l2) :
    dh_result (join l1 l2)
```

Question 3. Consider the following code, implementing the Noise message $\rightarrow es, ss$, modeling a mix of Alice's ephemeral key and Bob's static key, followed by a mix of Alice's and Bob's static keys.

```
let ck0: chaining_key public = init in
// es
let dh_es = dh e rs in
let ck1 = kdf ck0 dh_es in
// ss
let dh_ss = dh s rs in
let ck2 = kdf ck1 dh_ss
```

We consider two types of security labels:

- data with label [P "Alice"] corresponds to static data that can only be read by principal "Alice"
- data with label [S "Bob" sid] corresponds to ephemeral data that can only be read by principal "Bob" at session sid

Assuming we are currently at session sn, and using the signature for kdf below, give the types, including labels, associated to dh_es, dh_ss, ck1, and ck2.

```
val kdf (l1 l2: label) (k: chaining_key l1) (dh: dh_result l2) : chaining_key (meet l1 l2)
```

3 Stateful Verification

Question 1. Consider the following F* code, implementing a stateful sum:

```
let sum_tot (n:nat) = ((n+1) ∗ n) / 2

let rec sum_st' (r:ref nat) (n:nat)
  : ST unit (requires (λ _ → ⊤)) (ensures (λ _ _ _ → ⊤))
= if n > 0 then (r := !r + n; sum_st' r (n − 1))

let rec sum_st (n:nat)
```

```
  : ST nat (requires (λ _ → ⊤))
           (ensures (λ h0 x h1 → x == sum_tot n ∧ modifies !{} h0 h1))
= let r = alloc 0 in
  sum_st' r n;
  !r
```

Extend the signature of the intermediate function sum_st' so that sum_st typechecks.

Question 2. Consider the following F⋆ code:

```
let incr (r: ref int) : ST unit (requires (λ _ → ⊤))
            (ensures λh0 _ h1 → modifies !{r} h0 h1 ∧ sel h1 r == sel h0 r + 1)
  = r := !r + 1

let f (r1 r2: ref int) : ST unit (requires λ_ → ⊤))
            (ensures λh0 _ h1 → modifies !{r1, r2} h0 h1 ∧
                                sel h1 r1 == sel h0 r1 + 2 ∧ sel h1 r2 == sel h0 r2 + 1)
  = incr r1; incr r2; incr r1
```

Assuming that the implementation of incr satisfies its specification, provide a detailed proof of the correctness of f, indicating the known logical facts in each intermediate state (after the first, second and third function calls)

A Typing Rules for Information-Flow Control

Int
$$\gamma \vdash n : \tau$$

Var
$$\frac{\gamma(x) = \tau\ var}{\gamma \vdash x : \tau\ var}$$

Arith
$$\frac{\gamma \vdash e : \tau \qquad \gamma \vdash e' : \tau}{\gamma \vdash e + e' : \tau}$$

Assign
$$\frac{\gamma \vdash e : \tau\ var \qquad \gamma \vdash e' : \tau}{\gamma \vdash e := e' : \tau\ cmd}$$

Compose
$$\frac{\gamma \vdash c : \tau\ cmd \qquad \gamma \vdash c' : \tau\ cmd}{\gamma \vdash c; c' : \tau\ cmd}$$

R-val
$$\frac{\gamma \vdash e : \tau\ var}{\gamma \vdash e : \tau}$$

If
$$\frac{\gamma \vdash e : \tau \qquad \gamma \vdash c : \tau\ cmd \qquad \gamma \vdash c' : \tau\ cmd}{\gamma \vdash \textbf{if}\ e\ \textbf{then}\ c\ \textbf{else}\ c' : \tau\ cmd}$$

While
$$\frac{\gamma \vdash e : \tau \qquad \gamma \vdash c : \tau\ cmd}{\gamma \vdash \textbf{while}\ e\ \textbf{do}\ c : \tau\ cmd}$$

Base
$$\frac{\tau \leq \tau'}{\vdash \tau \subseteq \tau'}$$

Subtype
$$\frac{\gamma \vdash p : \rho \qquad \vdash \rho \subseteq \rho'}{\gamma \vdash p : \rho'}$$

Cmd-
$$\frac{\vdash \tau \subseteq \tau'}{\vdash \tau'\ cmd \subseteq \tau\ cmd}$$

B Typing Rules for Speculative Constant-Time

$$\textsc{Var}\quad \Gamma \vdash x : \Gamma(x)$$

$$\textsc{Op}\quad \frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash op(e_1, e_2) : \tau_1 \cup \tau_2}$$

$$\textsc{Const}\quad \Gamma \vdash n : (L, L)$$

$$\textsc{Sub}\quad \frac{\Gamma \vdash e : \tau \qquad \tau \leq \tau'}{\Gamma \vdash e : \tau'}$$

$$\textsc{If}\quad \frac{\Gamma \vdash b : (L, L) \qquad \Sigma_{|b}, \Gamma \vdash c_1 : \Sigma_1, \Gamma_1 \qquad \Sigma_{|!b}, \Gamma \vdash c_2 : \Sigma_2, \Gamma_2}{\Sigma, \Gamma \vdash \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \Sigma_1 \cap \Sigma_2, \Gamma_1 \cup \Gamma_2}$$

$$\textsc{Load}\quad \frac{\Gamma \vdash i : (L, L) \qquad \Gamma(a) = (\tau_n, \tau_s)}{\Gamma \vdash x = a[i] : \Gamma[x \leftarrow (\tau_n, H)]}$$

$$\textsc{Load}\quad \frac{\Gamma \vdash i : (L, L) \qquad \Gamma(a) = (\tau_n, \tau_s)}{\Sigma, \Gamma \vdash x = a[i] : \Sigma, \Gamma[x \leftarrow (\tau_n, H)]}$$

$$\textsc{Const-Load}\quad \frac{\text{n is constant}}{\Sigma, \Gamma \vdash x = a[n] : \Sigma, \Gamma[x \leftarrow \Gamma(a)]}$$

$$\textsc{Store}\quad \frac{\Gamma \vdash i : (L, L) \qquad \Gamma \vdash e : \tau \qquad \tau \leq \Gamma(a) \qquad \forall a' : \mathbf{A}, a' \neq a. \Gamma'[a'] = (\Gamma_n[a'], \tau_s \cup \Gamma_s[a'])}{\Sigma, \Gamma \vdash a[i] = e : \Sigma, \Gamma'}$$

$$\textsc{Seq}\quad \frac{\Sigma_0, \Gamma_0 \vdash c_1 : \Sigma_1, \Gamma_1 \qquad \Sigma_1, \Gamma_1 \vdash c_2 : \Sigma_2, \Gamma_2}{\Sigma_0, \Gamma_0 \vdash c_1;\ c_2 : \Sigma_2, \Gamma_2}$$

$$\textsc{Assign}\quad \frac{\Gamma \vdash e : \tau}{\Sigma, \Gamma \vdash x = e : \Sigma, \Gamma[x \leftarrow \tau]}$$

$$\textsc{Set-MS}\quad \mathsf{ms}_{|e}, \Gamma \vdash \mathsf{ms} = \mathrm{set\_ms}(e) : \mathsf{ms}, \Gamma$$

$$\textsc{Protect}\quad \frac{\Gamma' = \Gamma[y \leftarrow (\Gamma_n(x), \Gamma_n(x))]}{\mathsf{ms}, \Gamma \vdash y = \mathrm{protect}(x, \mathsf{ms}) : \mathsf{ms}, \Gamma'}$$