SOLUTION NOTES

Security (Part II) 2002 Paper 7 Question 6 (MGK)

(*a*) (*i*) Even though a message authentication code would prevent an attacker from placing a completely new record onto the stack, it will not prevent an attacker from exchanging records with ones that have been put there previously (replay and reordering attack).

(*ii*) The basic idea of one solution is that we keep in an on-card variable A the hash of the top record on the external stack and each record on the external stack contains a hash of the record below it. The bottom record is either marked specially, for instance with the value zero in the field that would otherwise contain the hash value of the next lower record, or we keep in the card a counter C of how many records there are on the stack.

On-card data:

$$A = h(Y_3)$$

External Stack:

$$\begin{aligned} Y_3 &= (X_3, h(Y_2)) \\ Y_2 &= (X_2, h(Y_1)) \\ Y_1 &= (X_1, 0) \end{aligned}$$

[There are several alternative solutions. One involves attaching to an externally stored record both its height on the stack as well as a `push` counter, all together protected by a MAC.]

(*iii*)

```
secure_init() {
  A = 0;
  C = 0;
  init();
}

secure_push(X) {
  push(X, A);
  A = h(X, A);
  C = C + 1;
}

secure_isempty() {
  return (C == 0);
}
```

```
secure_pop() {
   (X, A') = pop();
   C = C - 1;
   if (h(X, A') == A) then
      A = A';
      return X;
   else
      raise alarm;
}
```

(*b*) Keep in the card a counter A for the number of records that have been placed in the queue so far and a counter B for the number of records that have been removed from it. Store counter value A with each record placed under MAC protection into the queue and check for each removed record that the counter value in it equals B.

[*This question refers mainly to the course chapters on "message authentication codes", "hash functions", "symmetric cryptographic protocols" and to some degree also "hardware security".*]