```
(* Bitlocker protocol.

Found in "Formal analysis of protocols based on TPM state registers."
by Stéphanie Delaune, Steve Kremer, Mark D. Ryan, and Graham Steel
in Proceedings of the 24th IEEE Computer Security Foundations Symposium
(CSF'11), pp. 66–82, IEEE Computer Society Press, Cernay-la-Ville, France, June
2011.
*)

(********************************
***     TPM Declaration     ***
*********************************)

free c:channel.
free pcr:channel [private].

type type_key.

free bindk:type_key.
free sealk:type_key.

type private_key.
type public_key.

type value_pcr.
free nil:value_pcr.

(* Functions of the TPM *)

fun hpcr(value_pcr,bitstring) : value_pcr.
free init : value_pcr.

fun pk(private_key) : public_key.
fun certPCR(private_key,value_pcr,bitstring) : bitstring.
fun certKey(private_key,public_key,value_pcr) : bitstring.

reduc forall sk:private_key, v:value_pcr, d:bitstring ;
check_certPCR(certPCR(sk,v,d),pk(sk)) = (v,d).
reduc forall sk:private_key, v:value_pcr, xpk:public_key ;
check_certKey(certKey(sk,xpk,v),pk(sk)) = (xpk,v).

fun aenc(public_key,bitstring) : bitstring.
reduc forall sk:private_key, d:bitstring; adec(sk,aenc(pk(sk),d)) = d.

free aik:private_key [private].
free srk:private_key [private].
table keyloaded(private_key,public_key,type_key,value_pcr).

fun wrap(public_key,private_key,type_key,bitstring,value_pcr) : bitstring.
```

```
reduc forall x_pk:public_key, x_key:private_key, t_key:type_key, data:bitstring,
x_pcr:value_pcr;
 unwrap(wrap(x_pk,x_key,t_key,data, x_pcr)) = (x_pk,x_key,t_key,data,x_pcr)
[private].

fun seal(public_key,bitstring,bitstring,value_pcr) : bitstring.
reduc forall x_pk:public_key, d:bitstring, p:bitstring, v:value_pcr;
unseal(seal(x_pk,d,p,v)) = (x_pk,d,p,v) [private].

(*********************************
***     TPM Functionality     ***
**********************************)

(* The commands *)
free load: bitstring.
free read: bitstring.
free quote: bitstring.
free wrap_key : bitstring.
free certify : bitstring.
free unbind : bitstring.
free seal_data: bitstring.
free unseal_data: bitstring.
free extend : bitstring.
free reboot : bitstring.
free tpm_proof: bitstring [private].

(* Read the value of the PCR *)
let Read =
 in(c,=read);
 in(pcr,v:value_pcr);
 out(pcr,v);
 out(c,v).

(* Generate a certificate of an input value. *)
let Quote =
 in(c,(=quote,x:bitstring));
 in(pcr,v:value_pcr);
 out(pcr,v);
 out(c,certPCR(aik,v,x)).

(* Create Wrap Key *)
let CreateWrapKey =
 in(c,(=wrap_key,x_pk:public_key,t:type_key,v_lock:value_pcr));
 in(pcr,v_cur:value_pcr);
 out(pcr,v_cur);
 get keyloaded(x_key:private_key,=x_pk,t':type_key,v:value_pcr) in
 if v = nil || v = v_cur then
 new key[v_cur,v_lock]:private_key;
 out(c, (pk(key),wrap(x_pk,key,t,tpm_proof,v_lock))).
```

```

(* Load wrapped key *)
let LoadKey2 =
  in(c,(=load,x_pk:public_key,x_w:bitstring));
  let (y_pk:public_key,x_key:private_key,t:type_key,=tpm_proof,x_pcr:value_pcr) =
unwrap(x_w) in
  if pk(x_key) = x_pk then
  in(pcr,v:value_pcr);
  out(pcr,v);
  get keyloaded(x_sk:private_key,=y_pk,t':type_key,v':value_pcr) in
  if v = v' || v' = nil then
  insert keyloaded(x_key,x_pk,t,x_pcr).

(* Certify Key *)
let CertifyKey =
  in(c,(=certify,x_pk:public_key));
  get keyloaded(x_key:private_key,=x_pk,t:type_key,v:value_pcr) in
  out(c,certKey(aik,x_pk,v)).

(* Unbind *)
let Unbind =
  in(c,(=unbind, x_pk:public_key, cypher:bitstring));
  in(pcr,v:value_pcr);
  out(pcr,v);
  get keyloaded(x_sk:private_key,=x_pk,=bindk,v':value_pcr) in
  if v' = nil || v = v' then
  out(c,adec(x_sk,cypher)).

(* Seal *)
let Seal =
  in(c,(=seal_data, d:bitstring, x_pcr:value_pcr, x_pk:public_key));
  in(pcr,v:value_pcr);
  out(pcr,v);
  get keyloaded(x_sk:private_key,=x_pk,=sealk,v':value_pcr) in
  if v' = nil || v = v' then
  out(c,seal(x_pk, d, tpm_proof, x_pcr)).

(* Unseal *)
let Unseal =
  in(c,(=unseal_data, x:bitstring));
  let (x_pk:public_key,d:bitstring,=tpm_proof,v':value_pcr) = unseal(x) in
  in(pcr,v:value_pcr);
  out(pcr,v);
  get keyloaded(x_sk:private_key,=x_pk,=sealk,v'':value_pcr) in
  if (v' = nil && v'' = nil) || (v' = nil && v = v'') || (v' = v && v'' = nil) || (v' = v && v''
= v) then
  out(c,d).

(* Extend *)
```

```
let Extend =
  in(c,(=extend, x:bitstring));
  in(pcr,v:value_pcr);
  out(pcr,hpcr(v,x)).

let Initialisation =
  insert keyloaded(srk,pk(srk),bindk,nil) | out(c, pk(srk)).

let Main_TPM =
  Initialisation | ! (Read | Quote | CreateWrapKey | LoadKey2 | CertifyKey |
Unbind | Seal | Unseal | Extend).

free deny:bitstring.

(** Alice role **)

free vmk:bitstring [private].
free bios:bitstring.
free loader:bitstring.

fun abs_secret(bitstring,bitstring):bitstring [private].

let Alice =
  out(c,(wrap_key,pk(srk),sealk,nil));
  in(c,(x_pk:public_key,w:bitstring));
  out(c,(load,x_pk,w));
  get keyloaded(x_sk:private_key,=x_pk,=sealk,v':value_pcr) in
  out(c,seal(x_pk,vmk,tpm_proof,hpcr(hpcr(init,bios),loader))).

let reboot_and_measure_BIOS_and_loader =
  in(c,(x_bios:bitstring,x_loader:bitstring));
  in(pcr,v:value_pcr);
  if x_bios = bios && x_loader = loader
  then out(pcr,hpcr(hpcr(hpcr(init,x_bios),x_loader),deny))
  else if x_bios = bios
  then out(pcr,hpcr(hpcr(init,bios),x_loader))
  else out(pcr,hpcr(init,x_bios)).

let first_boot_measure_BIOS_and_loader =
  in(c,(x_bios:bitstring,x_loader:bitstring));
  if x_bios = bios && x_loader = loader
  then out(pcr,hpcr(hpcr(hpcr(init,x_bios),x_loader),deny))
  else if x_bios = bios
  then out(pcr,hpcr(hpcr(init,bios),x_loader))
  else out(pcr,hpcr(init,x_bios)).

let Main_Process =
    (! (Alice | reboot_and_measure_BIOS_and_loader) ) | Main_TPM |
first_boot_measure_BIOS_and_loader.
```

```

query attacker(vmk).

process Main_Process  | ! in(pcr,x:value_pcr); out(pcr,x)

```