

Solution of Exercise Sheet 5

Out: Thu, Jan 14, 2010

Due: Thu, Jan 28, 2010, 10am

Problem 1: An automated proof in CryptoVerif (10 points)

Use CryptoVerif to produce an automated proof of the following fact: If f is a one-way permutation, then $g := f \circ f$ is a one-way function.

As a help, I provide the following example input file for CryptoVerif. It shows all the necessary syntactic elements, but both the goal and the user-defined facts have nothing to do with one-way functions. (Actually, the input file does not make much sense.)

```
(* Channel names need to be declared. *)
```

```
channel start,adv.
```

```
(* Events need to be declared *)
```

```
event bad.
```

```
(* Type declarations. [fixed] means that it is possible to uniformly  
pick random values of this type *)
```

```
type D [fixed].
```

```
type T [fixed].
```

```
(* Parameters. Needed for replications (!n) *)
```

```
param n.
```

```
(* Function declaration. Example: f takes one argument of type D and  
returns a value of type T. g takes two arguments of type T and returns  
a value of type T. *)
```

```
fun f(D):T.
```

```
fun g(T,T):T.
```

```
(* Some user-defined equation that we claim to hold *)
```

```
forall x:D, x':D; g(g(f(x),f(x)),g(f(x'),f(x'))) = g(f(x),f(x')).
```

```
(* Declaration of a negligible probability *)
```

```
proba negl.
```

```
(* Some crypto assumption *)
```

```
equiv
```

```

!n new x: T; (( () -> x, (x':T) -> g(x,x'))
<=(negl)=>
!n new x: D; (( () -> f(x), (x':T) -> g(f(x),x'))).

```

query event bad ==> false.

```

process
  in(start,());
  new x : D;
  let y = f(x) in
  out(adv,g(y,y));
  in(adv,x':T);
  if g(f(x),x')=g(x',x') then event bad

```

Solution.

```

(* Channel names need to be declared. *)
channel start,adv.

```

```

(* Events need to be declared *)
event bad.

```

```

(* Type declarations. D represents range and domain of the one-way
permutation f. [fixed] means that it is possible to uniformly pick
random values of this type *)
type D [fixed].

```

```

(* Parameters. Needed for replications (!n) *)
param n.

```

```

(* Function declarations. Both f and g are functions from D to D. *)
fun f(D):D.
fun g(D):D.

```

```

(* Injectivity of f. *)
forall x:D, x':D; (f(x)=f(x')) = (x=x').

```

```

(* Declaration of a negligible probability *)
proba negl.

```

```

(* One-wayness of f *)
equiv
!n new x: D; (( () -> f(x), (x':D) -> x=x')
<=(negl)=>

```

```

!n new x: D; (( () -> f(x), (x':D) -> false).

(* Definition of g *)
forall x:D; g(x) = f(f(x)).

query event bad ==> false.

(* One-wayness game for g *)
process
  in(start, ());
  new x : D;
  out(adv, g(x));
  in(adv, x':D);
  if g(x)=g(x') then event bad

```

.noitulor

Note: You get 1.23 extra points if you come up with functions f and g that satisfy the user-defined facts given in the example CryptoVerif input above.

Solution. Let the types D and T represent the same set. Let f be the identity and g be a constant function. Then f and g satisfy the user-defined facts. Note, however, that for these choices of f and g , the process given at the end of the input file raises the event bad .

.noitulor

Note: You can download CryptoVerif at <http://www.cryptoverif.ens.fr/>. Alternatively, CryptoVerif has been installed on the Computer Science CIP pool in the directory `/home/stud/kimpec/cryptoverif`. Note that to run CryptoVerif, you have to make `/home/stud/kimpec/cryptoverif` the current directory. The above example input file is provided in `/home/stud/kimpec/cryptoverif/example.cv`. A manual for CryptoVerif can be found in `/home/stud/kimpec/cryptoverif/docs/manual.pdf`.