

SQUIRREL

An Interactive Prover for Protocol Verification in the Computational Model

D. Baelde, S. Delaune, C. Jacomme, A. Koutsos & S. Moreau



We illustrate our approach on the Basic Hash protocol which is an RFID protocol involving multiple tags and readers. Each tag stores a secret key that is never updated, and the readers have access to a shared database containing all these keys. The protocol is as follows:

$$T \rightarrow R : \langle n, H(n, key) \rangle$$

Here, n is a fresh name and key is the secret key. When receiving a message, the reader checks that it is a pair whose second component is a hash of the first component using one of the keys from the database. We illustrate how SQUIRREL works on a simple authentication goal. The screenshots below show how the proof can be done step by step.

Poster Demo

Hopefully, more online with the screen sharing feature!

```
basic-hash.sp
[goal> Focused goal (1/1):
System: default/both
-----
forall (j:index),
(happens(R(j)) =>
(cond@R(j) =>
exists (i,k:index),
((T(i,k) < R(j)) && fst(output@T(i,k)) = fst(input@R(j))) &&
snd(output@T(i,k)) = snd(input@R(j))))))

hash h
abstract ok : message
abstract ko : message
name key : index->message
channel cT
channel cR

process tag(i:index,k:index) =
new nT;
out(cT, <nT, h(nT,key(i))>)

process reader(j:index) =
in(cT,x);
if exists (i,k:index), snd(x) = h(fst(x),key(i)) then
out(cR,ok)
else
out(cR,ko)

system ((!_j R: reader(j)) | (!_i !_k T: tag(i,k))).
(* Authentication goal for the action R (then branch of the reader) *)

goal wa_R :
forall (j:index),
happens(R(j)) =>
(cond@R(j) =>
exists (i,k:index), T(i,k) < R(j) &&
fst(output@T(i,k)) = fst(input@R(j)) &&
snd(output@T(i,k)) = snd(input@R(j)))).

Proof.
intro *.
expand cond@R(j).
euf Meq.
exists i, k0.
Qed.
```

```
basic-hash.sp
[goal> Focused goal (1/1):
System: default/both
Variables: i,j,k,k0:index
CtT: T(i,k0) < R(j)
Hap: happens(R(j))
Meq: snd(input@R(j)) = h(fst(input@R(j)),key(i))
Meq0: nT(i,k0) = fst(input@R(j))
-----
exists (i,k:index),
((T(i,k) < R(j)) && fst(output@T(i,k)) = fst(input@R(j))) &&
snd(output@T(i,k)) = snd(input@R(j)))

hash h
abstract ok : message
abstract ko : message
name key : index->message
channel cT
channel cR

process tag(i:index,k:index) =
new nT;
out(cT, <nT, h(nT,key(i))>)

process reader(j:index) =
in(cT,x);
if exists (i,k:index), snd(x) = h(fst(x),key(i)) then
out(cR,ok)
else
out(cR,ko)

system ((!_j R: reader(j)) | (!_i !_k T: tag(i,k))).
(* Authentication goal for the action R (then branch of the reader) *)

goal wa_R :
forall (j:index),
happens(R(j)) =>
(cond@R(j) =>
exists (i,k:index), T(i,k) < R(j) &&
fst(output@T(i,k)) = fst(input@R(j)) &&
snd(output@T(i,k)) = snd(input@R(j)))).

Proof.
intro *.
expand cond@R(j).
euf Meq.
exists i, k0.
Qed.
```

```
*response*
Un-processed system:
[!_j R: reader(j)] | (!_i !_k T: tag(i,k))

Processed system:
[!_j
in(cT,x);
if exists (i,k:index), snd(x) = h(fst(x),key(i)) then
R: out(cR,ok); null else R1: out(cR,ko); null] |
(!_i !_k T: out(cT,pair(nT(i,k),h(nT(i,k),key(i))))); null)

System default registered with actions (init,R,R1,T).

hash h
abstract ok : message
abstract ko : message
name key : index->message
channel cT
channel cR

process tag(i:index,k:index) =
new nT;
out(cT, <nT, h(nT,key(i))>)

process reader(j:index) =
in(cT,x);
if exists (i,k:index), snd(x) = h(fst(x),key(i)) then
out(cR,ok)
else
out(cR,ko)

system ((!_j R: reader(j)) | (!_i !_k T: tag(i,k))).
(* Authentication goal for the action R (then branch of the reader) *)

goal wa_R :
forall (j:index),
happens(R(j)) =>
(cond@R(j) =>
exists (i,k:index), T(i,k) < R(j) &&
fst(output@T(i,k)) = fst(input@R(j)) &&
snd(output@T(i,k)) = snd(input@R(j)))).

Proof.
intro *.
expand cond@R(j).
euf Meq.
exists i, k0.
Qed.
```

```
basic-hash.sp
[goal> Focused goal (1/1):
System: default/both
Variables: j:index
H: cond@R(j)
Hap: happens(R(j))
-----
exists (i,k:index),
((T(i,k) < R(j)) && fst(output@T(i,k)) = fst(input@R(j))) &&
snd(output@T(i,k)) = snd(input@R(j)))

hash h
abstract ok : message
abstract ko : message
name key : index->message
channel cT
channel cR

process tag(i:index,k:index) =
new nT;
out(cT, <nT, h(nT,key(i))>)

process reader(j:index) =
in(cT,x);
if exists (i,k:index), snd(x) = h(fst(x),key(i)) then
out(cR,ok)
else
out(cR,ko)

system ((!_j R: reader(j)) | (!_i !_k T: tag(i,k))).
(* Authentication goal for the action R (then branch of the reader) *)

goal wa_R :
forall (j:index),
happens(R(j)) =>
(cond@R(j) =>
exists (i,k:index), T(i,k) < R(j) &&
fst(output@T(i,k)) = fst(input@R(j)) &&
snd(output@T(i,k)) = snd(input@R(j)))).

Proof.
intro *.
expand cond@R(j).
euf Meq.
exists i, k0.
Qed.
```

```
basic-hash.sp
[goal> Goal wa_R is proved

hash h
abstract ok : message
abstract ko : message
name key : index->message
channel cT
channel cR

process tag(i:index,k:index) =
new nT;
out(cT, <nT, h(nT,key(i))>)

process reader(j:index) =
in(cT,x);
if exists (i,k:index), snd(x) = h(fst(x),key(i)) then
out(cR,ok)
else
out(cR,ko)

system ((!_j R: reader(j)) | (!_i !_k T: tag(i,k))).
(* Authentication goal for the action R (then branch of the reader) *)

goal wa_R :
forall (j:index),
happens(R(j)) =>
(cond@R(j) =>
exists (i,k:index), T(i,k) < R(j) &&
fst(output@T(i,k)) = fst(input@R(j)) &&
snd(output@T(i,k)) = snd(input@R(j)))).

Proof.
intro *.
expand cond@R(j).
euf Meq.
exists i, k0.
Qed.
```

```
basic-hash.sp
Retract (undo) whole buffer

hash h
abstract ok : message
abstract ko : message
name key : index->message
channel cT
channel cR

process tag(i:index,k:index) =
new nT;
out(cT, <nT, h(nT,key(i))>)

process reader(j:index) =
in(cT,x);
if exists (i,k:index), snd(x) = h(fst(x),key(i)) then
out(cR,ok)
else
out(cR,ko)

system ((!_j R: reader(j)) | (!_i !_k T: tag(i,k))).
(* Authentication goal for the action R (then branch of the reader) *)

goal wa_R :
forall (j:index),
happens(R(j)) =>
(cond@R(j) =>
exists (i,k:index), T(i,k) < R(j) &&
fst(output@T(i,k)) = fst(input@R(j)) &&
snd(output@T(i,k)) = snd(input@R(j)))).

Proof.
intro *.
expand cond@R(j).
euf Meq.
exists i, k0.
Qed.
```

```
basic-hash.sp
[goal> Focused goal (1/1):
System: default/both
Variables: i,j,k:index
Meq: snd(input@R(j)) = h(fst(input@R(j)),key(i))
-----
exists (i,k:index),
((T(i,k) < R(j)) && fst(output@T(i,k)) = fst(input@R(j))) &&
snd(output@T(i,k)) = snd(input@R(j)))

hash h
abstract ok : message
abstract ko : message
name key : index->message
channel cT
channel cR

process tag(i:index,k:index) =
new nT;
out(cT, <nT, h(nT,key(i))>)

process reader(j:index) =
in(cT,x);
if exists (i,k:index), snd(x) = h(fst(x),key(i)) then
out(cR,ok)
else
out(cR,ko)

system ((!_j R: reader(j)) | (!_i !_k T: tag(i,k))).
(* Authentication goal for the action R (then branch of the reader) *)

goal wa_R :
forall (j:index),
happens(R(j)) =>
(cond@R(j) =>
exists (i,k:index), T(i,k) < R(j) &&
fst(output@T(i,k)) = fst(input@R(j)) &&
snd(output@T(i,k)) = snd(input@R(j)))).

Proof.
intro *.
expand cond@R(j).
euf Meq.
exists i, k0.
Qed.
```

