

## Output Transcript of ProVerif Analyses

Output for the script TID-RID-Privacy.pv:

Process:

```
(
  {1}!
  {2}new imsi_ms: ident;
  {3}new ki: key;
  {4}insert keys(imsi_ms,ki);
  {5}out(pubChannel, (ID,imsi_ms));
  {6}in(pubChannel, (=CHALLENGE,masked_rid_ms:
bitstring,masked_tid_ms: bitstring,r_ms: nonce,enc_sqn_ms:
bitstring,mac_ms: mac));
  {7}let ak_ms: anonymityKey =
f5(masked_rid_ms,masked_tid_ms,r_ms,ki) in
  {8}let sqn_ms: bitstring = adecrypt(enc_sqn_ms,ak_ms) in
  {9}if (f1(sqn_ms,masked_rid_ms,masked_tid_ms,r_ms,ki) =
mac_ms) then
    {10}let res_ms: resp =
f2(masked_rid_ms,masked_tid_ms,r_ms,ki) in
    {11}let ck_ms: cipherKey =
f3(masked_rid_ms,masked_tid_ms,r_ms,ki) in
    {12}let ik_ms: integrityKey =
f4(masked_rid_ms,masked_tid_ms,r_ms,ki) in
    {13}event endMS(imsi_ms,ck_ms,ik_ms);
    {14}event begSN(imsi_ms,ck_ms,ik_ms);
    {15}out(pubChannel, (RES,res_ms))
  ) | (
    {16}!
    {17}in(pubChannel, (=ID,imsi_sn: ident));
    {18}out(secureChannel, (AV_REQ,imsi_sn));
    {19}in(secureChannel, (=AV,imsi_hn_sn:
ident,masked_rid_sn: bitstring,masked_tid_sn: bitstring,r_sn:
nonce,enc_sqn_sn: bitstring,mac_sn: mac,xres_sn: resp,ck_sn:
cipherKey,ik_sn: integrityKey));
    {20}event begMS(imsi_hn_sn,ck_sn,ik_sn);
    {21}out(pubChannel,
(CHALLENGE,masked_rid_sn,masked_tid_sn,r_sn,enc_sqn_sn,mac_sn)
);
    {22}in(pubChannel, (=RES,res_sn: resp));
    {23}if (res_sn = xres_sn) then
    {24}event endSN(imsi_hn_sn,ck_sn,ik_sn)
  ) | (
    {25}!
    {26}in(secureChannel, (=AV_REQ,imsi_hn: ident));
    {39}get keys(=imsi_hn,ki_hn: key) in
    {27}new r_hn: nonce;
    {28}let ek2_hn: maskKey = f53(sqn,ki_hn) in
    {29}let masked_rid_hn: bitstring = mencrypt(rid,ek2_hn) in
    {30}let ek_hn: maskKey = f52(sqn,ki_hn) in
```

```

    {31}let masked_tid_hn: bitstring = mencrypt(tid,ek_hn) in
    {32}let mac_hn: mac =
f1(sqn,masked_rid_hn,masked_tid_hn,r_hn,ki_hn) in
    {33}let xres_hn: resp =
f2(masked_rid_hn,masked_tid_hn,r_hn,ki_hn) in
    {34}let ck_hn: cipherKey =
f3(masked_rid_hn,masked_tid_hn,r_hn,ki_hn) in
    {35}let ik_hn: integrityKey =
f4(masked_rid_hn,masked_tid_hn,r_hn,ki_hn) in
    {36}let ak_hn: anonymityKey =
f5(masked_rid_hn,masked_tid_hn,r_hn,ki_hn) in
    {37}let enc_sqn_hn: bitstring = aencrypt(sqn,ak_hn) in
    {38}out(secureChannel,
(AV,imsi_hn,masked_rid_hn,masked_tid_hn,r_hn,enc_sqn_hn,mac_hn
,xres_hn,ck_hn,ik_hn))
)

```

```

-- Query event(endMS(x1,x2,x3)) ==> event(begMS(x1,x2,x3))
Completing...
Starting query event(endMS(x1,x2,x3)) ==>
event(begMS(x1,x2,x3))
goal reachable: begin(begMS(imsi_ms[!1 =
@sid_2166],f3(mencrypt(rid[],f53(sqn[],ki[!1 =
@sid_2166])),mencrypt(tid[],f52(sqn[],ki[!1 =
@sid_2166])),r_hn[ki_hn = ki[!1 = @sid_2166],imsi_hn =
imsi_ms[!1 = @sid_2166],!1 = @sid_2167],ki[!1 =
@sid_2166]),f4(mencrypt(rid[],f53(sqn[],ki[!1 =
@sid_2166])),mencrypt(tid[],f52(sqn[],ki[!1 =
@sid_2166])),r_hn[ki_hn = ki[!1 = @sid_2166],imsi_hn =
imsi_ms[!1 = @sid_2166],!1 = @sid_2167],ki[!1 = @sid_2166]))
-> end(endMS(imsi_ms[!1 =
@sid_2166],f3(mencrypt(rid[],f53(sqn[],ki[!1 =
@sid_2166])),mencrypt(tid[],f52(sqn[],ki[!1 =
@sid_2166])),r_hn[ki_hn = ki[!1 = @sid_2166],imsi_hn =
imsi_ms[!1 = @sid_2166],!1 = @sid_2167],ki[!1 =
@sid_2166]),f4(mencrypt(rid[],f53(sqn[],ki[!1 =
@sid_2166])),mencrypt(tid[],f52(sqn[],ki[!1 =
@sid_2166])),r_hn[ki_hn = ki[!1 = @sid_2166],imsi_hn =
imsi_ms[!1 = @sid_2166],!1 = @sid_2167],ki[!1 = @sid_2166]))
RESULT event(endMS(x1,x2,x3)) ==> event(begMS(x1,x2,x3)) is
true.

```

```

-- Query event(endSN(x1_2171,x2_2172,x3_2173)) ==>
event(begSN(x1_2171,x2_2172,x3_2173))
Completing...
Starting query event(endSN(x1_2171,x2_2172,x3_2173)) ==>
event(begSN(x1_2171,x2_2172,x3_2173))
goal reachable: begin(begSN(imsi_ms[!1 =
@sid_3826],f3(mencrypt(rid[],f53(sqn[],ki[!1 =
@sid_3826])),mencrypt(tid[],f52(sqn[],ki[!1 =
@sid_3826])),r_hn[ki_hn = ki[!1 = @sid_3826],imsi_hn =
imsi_ms[!1 = @sid_3826],!1 = @sid_3827],ki[!1 =
@sid_3826]),f4(mencrypt(rid[],f53(sqn[],ki[!1 =
@sid_3826])),mencrypt(tid[],f52(sqn[],ki[!1 =

```

```

@sid_3826])),r_hn[ki_hn = ki[!1 = @sid_3826],imsi_hn =
imsi_ms[!1 = @sid_3826],!1 = @sid_3827],ki[!1 = @sid_3826]))
-> end(endSN(imsi_ms[!1 =
@sid_3826],f3(mencrypt(rid[],f53(sqn[],ki[!1 =
@sid_3826])),mencrypt(tid[],f52(sqn[],ki[!1 =
@sid_3826])),r_hn[ki_hn = ki[!1 = @sid_3826],imsi_hn =
imsi_ms[!1 = @sid_3826],!1 = @sid_3827],ki[!1 =
@sid_3826]),f4(mencrypt(rid[],f53(sqn[],ki[!1 =
@sid_3826])),mencrypt(tid[],f52(sqn[],ki[!1 =
@sid_3826])),r_hn[ki_hn = ki[!1 = @sid_3826],imsi_hn =
imsi_ms[!1 = @sid_3826],!1 = @sid_3827],ki[!1 = @sid_3826]))
RESULT event(endSN(x1_2171,x2_2172,x3_2173)) ==>
event(begSN(x1_2171,x2_2172,x3_2173)) is true.

```

```

-- Query not attacker(rid[])
Completing...
Starting query not attacker(rid[])
RESULT not attacker(rid[]) is true.

```

```

-- Query not attacker(tid[])
Completing...
Starting query not attacker(tid[])
RESULT not attacker(tid[]) is true.

```

```

-- Query not attacker(sqn[])
Completing...
Starting query not attacker(sqn[])
RESULT not attacker(sqn[]) is true.

```

Output for the script Correctness-Recovery.pv:

File "WiSec/Correctness-Recovery.pv", line 99, character 6 -  
line 99, character 13:

Warning: identifier success rebound

File "WiSec/Correctness-Recovery.pv", line 92, character 7 -  
line 92, character 14:

Warning: identifier success rebound

Process:

```
(
  {1}!
  {2}new imsi_ms: ident;
  {3}new ki: key;
  {4}insert keys(imsi_ms,ki);
  {5}out(pubChannel, (ID,imsi_ms));
  {6}in(pubChannel, (=CHALLENGE,masked_rid_ms:
bitstring,masked_tid_ms: bitstring,r_ms: nonce,enc_sqn_ms:
bitstring,mac_ms: mac));
  {7}new d_mac: mac;
  {8}new d_rid: bitstring;
  {9}let ak_ms: anonymityKey =
f5(masked_rid_ms,masked_tid_ms,r_ms,ki) in
  {10}let sqn_ms: bitstring = adecrypt(enc_sqn_ms,ak_ms) in
  {11}if (f1(sqn_ms,masked_rid_ms,masked_tid_ms,r_ms,ki) =
mac_ms) then
    (
      {12}let res_ms: resp =
f2(masked_rid_ms,masked_tid_ms,r_ms,ki) in
      {13}let ck_ms: cipherKey =
f3(masked_rid_ms,masked_tid_ms,r_ms,ki) in
      {14}let ik_ms: integrityKey =
f4(masked_rid_ms,masked_tid_ms,r_ms,ki) in
      {15}let success_48: bool = true in
      {16}out(pubChannel,
(RES,success_48,res_ms,d_rid,d_mac))
    )
  else
    {17}new d_res: resp;
    {18}let success_49: bool = false in
    {19}let ek2_ms: maskKey = f53(sqn_ms,ki) in
    {20}let rid_ms: bitstring =
mdecrypt(masked_rid_ms,ek2_ms) in
    {21}let mac_m_ms: mac = f12(rid_ms,imsi_ms,ki) in
    {22}event begHN(imsi_ms,rid_ms);
    {23}out(pubChannel,
(RES,success_49,d_res,rid_ms,mac_m_ms))
  ) | (
    {24}!
    {25}in(pubChannel, (=ID,imsi_sn: ident));
    {26}out(secureChannel, (AV_REQ,imsi_sn));
    {27}in(secureChannel, (=AV,imsi_hn_sn:
ident,masked_rid_sn: bitstring,masked_tid_sn: bitstring,r_sn:
nonce,enc_sqn_sn: bitstring,mac_sn: mac,xres_sn: resp,ck_sn:
```

```

cipherKey,ik_sn: integrityKey));
  {28}out(pubChannel,
(CHALLENGE,masked_rid_sn,masked_tid_sn,r_sn,enc_sqn_sn,mac_sn)
);
  {29}in(pubChannel, (=RES,success_sn: bool,res_sn:
resp,rid_sn: bitstring,mac_m_sn: mac));
  {30}if (success_sn = true) then
  (
    {31}if (res_sn = xres_sn) then
      0
    )
  else
    {32}out(secureChannel,
(ERROR,imsi_hn_sn,rid_sn,mac_m_sn))
  ) | (
    {33}!
    {34}in(secureChannel, (=AV_REQ,imsi_hn: ident));
    {54}get keys(=imsi_hn,ki_hn: key) in
    {35}new r_hn: nonce;
    {36}new rid_hn: bitstring;
    {37}insert rids(rid_hn,imsi_hn);
    {38}let ek2_hn: maskKey = f53(sq_n,ki_hn) in
    {39}let masked_rid_hn: bitstring = mencrypt(rid_hn,ek2_hn)
in
    {40}let ek_hn: maskKey = f52(sq_n,ki_hn) in
    {41}let masked_tid_hn: bitstring = mencrypt(tid,ek_hn) in
    {42}let mac_hn: mac =
f1(sq_n,masked_rid_hn,masked_tid_hn,r_hn,ki_hn) in
    {43}let xres_hn: resp =
f2(masked_rid_hn,masked_tid_hn,r_hn,ki_hn) in
    {44}let ck_hn: cipherKey =
f3(masked_rid_hn,masked_tid_hn,r_hn,ki_hn) in
    {45}let ik_hn: integrityKey =
f4(masked_rid_hn,masked_tid_hn,r_hn,ki_hn) in
    {46}let ak_hn: anonymityKey =
f5(masked_rid_hn,masked_tid_hn,r_hn,ki_hn) in
    {47}let enc_sq_n_hn: bitstring = aencrypt(sq_n,ak_hn) in
    {48}out(secureChannel,
(AV,imsi_hn,masked_rid_hn,masked_tid_hn,r_hn,enc_sq_n_hn,mac_hn
,xres_hn,ck_hn,ik_hn));
    {49}in(secureChannel, (=ERROR,imsi_hn_s:
ident,rid_hn_2_50: bitstring,mac_m_hn: mac));
    {53}get rids(=rid_hn_2_50,imsi_hn2: ident) in
    {52}get keys(=imsi_hn2,ki_hn2: key) in
    {50}if (f12(rid_hn_2_50,imsi_hn2,ki_hn2) = mac_m_hn) then
    {51}event endHN(imsi_hn2,rid_hn_2_50)
  )
)

```

-- Query event(endHN(x1,x2)) ==> event(begHN(x1,x2))

Completing...

Starting query event(endHN(x1,x2)) ==> event(begHN(x1,x2))

goal reachable: begin(begHN(imsi\_ms[!1 =

```
@sid_2969],rid_hn[ki_hn = ki[!1 = @sid_2969],imsi_hn =  
imsi_ms[!1 = @sid_2969],!1 = @sid_2970])) ->  
end(endHN(imsi_ms[!1 = @sid_2969],rid_hn[ki_hn = ki[!1 =  
@sid_2969],imsi_hn = imsi_ms[!1 = @sid_2969],!1 = @sid_2970]))  
RESULT event(endHN(x1,x2)) ==> event(begHN(x1,x2)) is true.
```