Needham_schroeder

$I \rightarrow S$: I, R

$S \rightarrow I$: $\{PK_R, R\}_{SK(S)}$

$I \rightarrow R$: $\{N_i, I\}_{PK(R)}$

$R \rightarrow S$: R, I

$S \rightarrow R$: $\{PK(I), I\}_{SK(S)}$

$R \rightarrow I$: $\{N_i, N_R\}_{PK(I)}$

$I \rightarrow R$: $\{N_R\}_{PK(R)}$

Pour destribuer des clés public.

Problème

attack synchronised et agrement, car le clés que le serveur a evoyé elle peut être (replayed)

```
protocol exo1 (I, R, S)
{
 role I
 {
   fresh Ni : Nonce;
    var NR : Nonce;
    send_1 (I, S, (I, R));
    recv_2 (S, I, { PK(R), R}SK(S));
    send_3 (I, R, {Ni, I}PK(R));
    recv_6 (R, I, {Ni, Nr} PK(I));
    send_7 (I, R, {Nr} PK(R));
    claim_I1 (I, Secret, Ni);
    claim_I2 (I, Secret, Nr);
    claim_I3 (I, Nisynch);
 }
```

```
role R
{
 fresh Nr: Nonce;
 var Ni : Nonce;
 recv_3(I,R, {Ni,I}PK(R));
 send_4(R,S, (R,I));
 recv_5(S,R, {PK(I),I}SK(S));
 send_6(R,I, {Ni,Nr}PK(I));
 recv_7(I,R, {Nr}PK(R));
  claim_R1 (R, Secret,Nr);
  claim_R2(R, Secret, Ni);
} claim_R3(R,Nr synch);

role S
{
 recv_1 (I,S, (I,R));
  send_2 (S,I, {PK(R),R}SK(S));
  recv_4 (R,S, (R,I));
 send_5 (S,R, {PK(I),I}SK(S));
 }

}
```

$I \rightarrow R$; I

$R \rightarrow I$; N_R

$I \rightarrow R$: $\{N_R\}k(I,S)$

$R \rightarrow S$: $\{I, \{N_R\}k(I,S)\}k(R,S)$

$S \rightarrow R$: $\{N_R\}k(R,S)$;

protocol exo2' (I, R, S)

{

Role I

{

Var Nr: Nonce;

send_1(I, R, I);

recv_2(R, I, Nr);

send_3(I, R, {Nr}k(I,S));

}

$I \rightarrow R: I$

$R \rightarrow I: Nr$

$I \rightarrow R, \{Nr\}k(I,S))$

$S \rightarrow R: \{I, \{Nr\}k(I,S)\}k(R,S)$

role R

{

fresh Nr: Nonce;

Var T: Ticket);

recv_1(I, R, I);

send_2(R, I, Nr);

resc_3(I, R, T);

$\{Nr\}k(I,S)$.

```
send_4(R,S, {I, {Nr}k(I,S)} k(R,S));
recv_5(S,R, {Ni} k(R,S));
Claim_R3(R, Nsynch);
}
role S
{
Var Nr: Nonce;
recv_4(R,S, {I, {Ni}k(I,S)} k(R,S));
send_5(S,R, {Nr} k(R,S));
}
}
```

---

ex03: (Kaochow)

$I \to S: I, R, n_i$

$S \to R: T, \{I, R, n_i, k_{ir}\} k(R,S)$,

$R \to I: T$,

A → B : A, na

B → S : B, {A, na, nb} K(B,S)

S → A : {B, Kab, na, nb} K(A,S), {A, Kab} K(B,S)

A → B : {A, Kab} K(B,S) , {nb} Kab

B

protocol test 3 (A, B, S)

{

role A

{

fresh na : Nonce;

Var nb : Nonce;

Var ticket : Ticket;

Var Kab : Session key;

send_1 (A, B, na);

recv_3 (S, A, {B, Kab, na, nb} K(A,S), ticket);

send_4 (A, B, ticket , {nb} kab);

claim_5 (S, SKR, Kab);

}

role B

{

fresh nb; Nonce;

Var na : Nonce;

Var ticket : Ticket;

Var Kab : Session key;

recv_1 (A, B, na

send_2 (B, S, {A, na, nb} K(B,S));

recv_4 (A, B, {A, Kab} K(B,S), {nb} kab);

claim_6 (B, SKR, Kab);

```
role S
{
    fresh : Kab: Sessionkey;
    var Na, Nb: Nonce;
    recv_2 (B,S, B, {A,Na, Nb}K(B,S));
    send_3(S,A, {B,Kab, Na, Nb}K(A,S),
           {A, Kab} K(B,S));
}
```

yahalom_bam_pulson. /adversary

EXO 2

```
protocol test (S, R)
{
  roles
  {
    fresh data: Data;
    send_1 (S, R, {data}k(S,R));
    Claim_S1(S, Secret, data);
    Claim_S2 (S, Niagree);
    Claim_S3(S, Nisynch);
    Claim_S4(S, Alive);
  }
  role R
  {
    var rdata: Data;
    recv_1 (S, R, {rdata}k(S,R));
    Claim_R1 (R, Secret, rdata);
  }
}
```