```
%% PROTOCOL: Accès WIFI
%%
%% CLIENT_BORNE_SERVEUR :
%%
%% 01. C  <--- {CertificatServeur.SSID} --- B --- --- S
%% 02. C  --- {addrMAC.nonceClient.pkC}_pkS ---> B --- --- S
%% 03. C  --- --- B --- {AddrMAC.nonceClient.pkC}_pkS ---> S
%% 04. C  --- --- B <--- {cleSession.nonceClient.nonceServeur}_pkC --- S
%% 05. C  <--- {cleSession.nonceClient.nonceServeur}_pkC --- B --- --- S
%% 06. C  --- {h(mdp).nonceServeur.addrMac}_cleSession ---> B --- --- S
%% 07. C  --- --- B --- {h(mdp).nonceServeur.addrMac}_cleSession ---> S
%%
%%    Cas où le mdp est valide :
%% 08a. C  --- --- B <--- {ok.addrIP.cleReseau}_cleSession --- S
%% 09a. C  <--- {ok.addrIP.cleReseau}_cleSession --- B --- --- S
%% 10a. C  --- {ok}_cleReseau ---> B --- --- S
%% 11a. C  --- --- B --- {ok}_cleReseau ---> S
%%
%%    Cas où le mdp est faux :
%% 08b. C  --- --- B <--- {faux}_cleSession --- S
%% 09b. C  <--- {faux}_cleSession --- B --- --- S
%%    Retour à l'étape 06.
%%
%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% définition du rôle Client
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role client (C, B, S: agent,
            PKc, PKs: public_key,
            SND, RCV: channel(dy))
played_by C def=

  local State, IdClient: nat,
        AddrMAC, NonceClient, MdpClient, SSID, CertificatServeur,
NonceServeur, AddrIP: text,
        CleSession, CleReseau : symmetric_key

  const ok: text

  init State:=0 /\
  %% ID unique permettant de s'authentifier au serveur
       IdClient:=1

  transition

    02.  State=0 /\ RCV(SSID'.CertificatServeur') =|>
            State':=1 /\
```

```
            %%Verification du Certificat avec un CA por valider la PKs à
partir du certificat
            AddrMAC':=new() /\
            NonceClient':=new() /\
            secret(NonceClient', nonceClient, {C,S}) /\
            %% On ajoute l'identite du client au message
            SND({IdClient.AddrMAC'.NonceClient'.PKc}_PKs)

    06.  State=1 /\ RCV({CleSession'.NonceClient'.NonceServeur'}_PKc) =|>
            State':=2 /\
            MdpClient':=new() /\
            secret(MdpClient', mdpClient, {C,S}) /\
            SND({IdClient.h(MdpClient').NonceServeur'}_CleSession')

    10.  State=2 /\ RCV({ok.AddrIP'.CleReseau'}_CleSession) =|>
            State':=3 /\
            SND(IdClient.{ok}_CleReseau')



end role


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% définition du rôle Borne, initiant le protocole
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role borne (C, B, S: agent,
            PKc, PKs: public_key,
            SND, RCV: channel(dy))
played_by B def=

  local State, IdClient: nat,
        AddrMAC, NonceClient, MdpClient, SSID, CertificatServeur,
NonceServeur, AddrIP: text,
        CleSession, CleReseau : symmetric_key

  init State:=0

  transition

    01.  State=0 /\ RCV(start) =|>
            State':=1 /\
            SSID':=new() /\
            CertificatServeur':=new() /\
            SND(SSID'.CertificatServeur')

    03.  State=1 /\ RCV({IdClient.AddrMAC'.NonceClient'.PKc}_PKs) =|>
            State':=2 /\
            SND({IdClient.AddrMAC'.NonceClient'.PKc}_PKs)

    05.  State=2 /\ RCV({CleSession'.NonceClient'.NonceServeur'}_PKc) =|>
```

```
            State':=3 /\
            request(S, C, c_s_CleSession, CleSession') /\
            SND({CleSession'.NonceClient'.NonceServeur'}_PKc)

    07.  State=3 /\ RCV({IdClient.h(MdpClient').NonceServeur'}_CleSession')
=|>
            State':=4 /\
            SND({IdClient.h(MdpClient').NonceServeur'}_CleSession')

    09.  State=4 /\ RCV({ok.AddrIP'.CleReseau'}_CleSession) =|>
            State':=5 /\
            SND({ok.AddrIP'.CleReseau'}_CleSession)

    11.  State=5 /\ RCV({IdClient.ok.AddrIP}_CleReseau') =|>
            State':=6 /\
            SND({ok.AddrIP}_CleReseau')

end role




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% définition du rôle Serveur
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role serveur (C, B, S: agent,
            PKc, PKs: public_key,
            SND, RCV: channel(dy),
            ClientsOnConnecting: nat set)
played_by S def=

  local State,IdClient: nat,
        NonceServeur, AddrIP, AddrMAC, NonceClient, MdpClient, MdpReseau:
text,
        CleSession, CleReseau : symmetric_key

  const ok, mdpReseau: text

  init State:=0

  transition

    4.  State=0 /\ RCV({IdClient.AddrMAC'.NonceClient'.PKc}_PKs) /\
        not(in(IdClient, ClientsOnConnecting)) =|> %% Verifie si c'est un
client connu par le serveur
            State':=1 /\
            ClientsOnConnecting' := cons(IdClient,ClientsOnConnecting) /\
            CleSession':=new() /\
            NonceServeur':=new() /\
            secret(NonceServeur', nonceServeur, {C,S}) /\
```

```
            secret(CleSession', cleSession, {C,S}) /\
            witness(C, S, c_s_CleSession, CleSession') /\
            SND({CleSession'.NonceClient'.NonceServeur'}_PKc)

    8.  State=1 /\ RCV({IdClient.h(MdpClient').NonceServeur'}_CleSession)
/\
        in(IdClient, ClientsOnConnecting) =|>
            State':=2 /\
            equal(h(MdpClient'),h(mdpReseau)) /\
            AddrIP':=new() /\
            CleReseau':=new() /\
            secret(CleReseau', cleReseau, {C,S}) /\
            SND({ok.AddrIP'.CleReseau'}_CleSession)

    12.  State=2 /\ RCV({IdClient.ok.AddrIP}_CleReseau') /\
        in(IdClient, ClientsOnConnecting)=|>
            State':=3 /\
            ClientsOnConnecting' := delete(IdClient, ClientsOnConnecting)




end role



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% définition du rôle Session
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role session(C, B, S: agent, PKc, PKs: public_key, ClientsOnConnecting: nat
set) def=

  local SC, RC, SB, RB, SS, RS: channel(dy)

  composition

        client(C,B,S,PKc,PKs,SC,RC) /\
        borne(C,B,S,PKc,PKs,SB,RB) /\
        serveur(C,B,S,PKc,PKs,SS,RS,ClientsOnConnecting)
end role


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% définition du Scenario
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
role environment() def=

    local ClientsOnConnecting: nat set

    const c, b, s: agent,
          pkc, pks, pki: public_key,
          cleSession, cleReseau : symmetric_key,
          c_s_CleSession, nonceClient, nonceServeur : protocol_id,
          mdpClient : text,
          h : hash_func

    %% Ensemble des ClientsOnConnecting actuellement en cours de connexion
    %% (Seul le serveur connait cet ensemble)
    init ClientsOnConnecting := {}

    intruder_knowledge = {c, b, s, pkc, pks, pki, inv(pki), h}


    composition

        session(c,b,s,pkc,pks,ClientsOnConnecting)

end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% définition des Propriétés à vérifier
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

goal
    secrecy_of mdpClient
    secrecy_of nonceClient
    secrecy_of nonceServeur
    secrecy_of cleSession
    secrecy_of cleReseau
    authentication_on c_s_CleSession
end goal


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% lancement du rôle principal
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

environment()
```