usertype string;

const req, shd, ccs: string;

const mastersecret, clientfinished, serverfinished, keyexpansionclient, keyexpansionserver, accesslevel, success: string;

hashfunction h, prf;

macro m=( SID, ni, nr, {CerR,pk(R)}sk(CA), req, shd, {CerI, pk(I)}sk(CA), {PMS}pk(R));

macro m1= {h( SID, ni, nr, {CerR,pk(R)}sk(CA), req, shd, {CerI, pk(I)}sk(CA), {PMS}pk(R))}sk(I);

macro ms= prf(PMS, mastersecret,ni, nr);

macro mc= ( {CerI, pk(I)}sk(CA), {PMS}pk(R));

macro m2= prf (ms, serverfinished, h(SID, ni, nr, {CerR,pk(R)}sk(CA), req, shd, mc, {h( SID, ni, nr, {CerR,pk(R)}sk(CA), req, shd, mc)}sk(I)));

macro clientkey=prf(ms, keyexpansionclient, ni, nr);

macro serverkey= prf(ms, keyexpansionserver, ni, nr);

protocol eaptls (I,R,CA)

{

role I

{

const SID, CerI, CerR,PMS: Data;

fresh ni: Nonce;

var nr:Nonce;

send\_1 (I, R, SID, ni);

recv\_4 (R, I, SID, nr, {CerR,pk(R)}sk(CA), req, shd);

send\_5 (I, CA, I);

recv\_6 (CA, I, {I, {CerI, pk(I)}pk(I)}sk(CA));

send\_7 (I, R, {CerI, pk(I)}sk(CA), {PMS}pk(R), ccs);

send\_8 (I,R , {h( SID, ni, nr, {CerR,pk(R)}sk(CA), req, shd, {CerI, pk(I)}sk(CA), {PMS}pk(R) )}sk(I));

recv\_9 (R, I, ccs, prf(ms, serverfinished, h(m, m1)));

match (m2, prf(ms, serverfinished, h(m, m1)));

recv\_10 (R, I, {accesslevel}serverkey);

send\_11 (I, R, {success}clientkey);

claim\_I1 (I, Secret, CerR);

claim\_I2 (I, Secret, CerI);

claim\_I3 (I, Secret, PMS);

claim\_I4 (I, Secret, ms);

claim\_I5 (I, Secret, clientkey);

claim\_I6 (I, Secret, serverkey);

claim\_I7 (I, Nisynch);

claim\_I8 (I, Niagree);

}

role R

{

const SID, CerI, CerR, PMS: Data;

fresh nr: Nonce;

var ni:Nonce;

recv\_1 (I, R, SID, ni);

send\_2 (R, CA, R);

recv\_3 (CA, R, {R, {CerR, pk(R)}pk(R)}sk(CA));

send\_4 (R, I, SID, nr, {CerR,pk(R)}sk(CA), req, shd);

recv\_7 (I, R, {CerI, pk(I)}sk(CA), {PMS}pk(R), ccs);

recv\_8 (I,R , {h( SID, ni, nr, {CerR,pk(R)}sk(CA), req, shd, {CerI, pk(I)}sk(CA), {PMS}pk(R) )}sk(I));

match(m1, {h( SID, ni, nr, {CerR,pk(R)}sk(CA), req, shd, {CerI, pk(I)}sk(CA), {PMS}pk(R))}sk(I));

send\_9 (R, I, ccs, prf(ms, serverfinished, h(m, m1)));

send\_10 (R, I, {accesslevel}serverkey);

recv\_11 (I, R, {success}clientkey);

claim\_R1 (R, Secret, CerR);

claim\_R2 (R, Secret, CerI);

claim\_R3 (R, Secret, PMS);

claim\_R4 (R, Secret, ms);

claim\_R5 (R, Secret, clientkey);

claim\_R6 (R, Secret, serverkey);

claim\_R7 (R, Nisynch);

claim\_R8 (R, Niagree);

}

role CA

{

const CerI, CerR: Data;

recv\_2 (R, CA, R);

send\_3 (CA, R, {R, {CerR, pk(R)}pk(R)}sk(CA));

recv\_5 (I, CA, I);

send\_6 (CA, I, {I, {CerI, pk(I)}pk(I)}sk(CA));

claim\_CA1 (CA, Secret, CerR);

claim\_CA2 (CA, Secret, CerI);

}

}



