PEAP original

hashfunction prf;

hashfunction h1;

hashfunction h;

usertype String;

const ccs, csu, shd, mastersecret, password : String;

const PEAPsuccess, accesslevel, clientfinished, serverfinished, peapkeyingmaterial,uiname,urname : String;

macro ms= prf (pms, mastersecret, (ni1, nr1));

macro finishedi= prf (ms, clientfinished, h1(SID, ni1, nr1, csu, {CerR, pk(R)}sk(CA), shd, {pms}pk(R)));

macro finishedr= prf(ms, serverfinished, h1(SID, ni1, nr1, csu, {CerR, pk(R)}sk(CA), shd, {pms}pk(R), finishedi));

macro ikey= prf ( ms, peapkeyingmaterial, (ni1, nr1));

macro skey= prf ( ms, peapkeyingmaterial, (ni1, nr1));

protocol peap1(I, R, CA)

{

role I

{

const SID, ID, CerR, pms: Data;

fresh ni1: Nonce;

var nr1, n: Nonce;

send\_1 (I, R, SID, ni1, csu);

recv\_4 (R, I, SID, nr1, csu, {CerR, pk(R)}sk(CA), shd);

send\_5 (I, R, {pms}pk(R), ccs);

send\_6 (I,R, finishedi);

recv\_7 (R, I, ccs,finishedr);

match ( prf(ms, serverfinished, h1(SID, ni1, nr1, csu, {CerR, pk(R)}sk(CA), shd, {pms}pk(R), finishedi)), finishedr);

send\_8 (I,R, {uiname}ikey);

recv\_9 (R, I, {ID, n, uiname}skey);

send\_10 (I, R, {ID, h(n, password), urname}ikey);

recv\_11 (R, I, {PEAPsuccess, accesslevel}skey);

claim\_i1 (I, Secret, CerR);

claim\_i2 (I, Secret, pms);

claim\_i3 (I, Secret, ms);

claim\_i4 (I, Secret, ikey);

claim\_i5 (I, Secret, skey);

claim\_i6 (I, Secret, password);

claim\_i1 (I, Niagree);

claim\_i2 (I, Nisynch);

}

role R

{

const SID, ID, CerR, pms: Data;

fresh nr1, n: Nonce;

var ni1: Nonce;

recv\_1 (I, R, SID, ni1, csu);

send\_2 (R, CA, R);

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

send\_4 (R, I, SID, nr1, csu, {CerR, pk(R)}sk(CA), shd);

recv\_5 (I, R, {pms}pk(R), ccs);

recv\_6 (I,R, finishedi);

match (prf (ms, clientfinished, h(SID, ni1, nr1, csu, {CerR, pk(R)}sk(CA), shd, {pms}pk(R))), finishedi);

send\_7 (R, I, ccs, finishedr);

recv\_8 (I,R, {uiname}ikey);

send\_9 (R, I, {ID, n, uiname}skey);

recv\_10 (I, R, {ID, h(n, password), urname}ikey);

match (h(n,password), h(n,password));

send\_11(R, I, {PEAPsuccess, accesslevel}skey);

claim\_r1 (R, Secret, CerR);

claim\_r2 (R, Secret, pms);

claim\_r3 (R, Secret, ms);

claim\_r4 (R, Secret, ikey);

claim\_r5 (R, Secret, skey);

claim\_r6 (R, Secret, password);

claim\_r1 (R, Niagree);

claim\_r2 (R, Nisynch);

}

role CA

{

const CerR: Data;

recv\_2(R, CA, R);

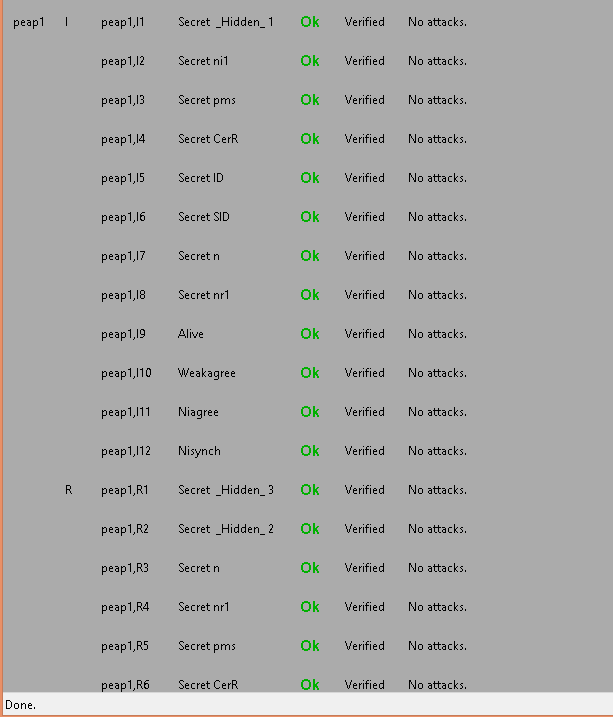
send\_3 (CA, R, {R, {CerR, pk(R)}pk(R)} sk(CA));

claim\_ca1 (CA, Secret, CerR);

}

}





PEAP modified

hashfunction prf;

hashfunction h1;

hashfunction h;

usertype String;

const ccs, csu, shd, mastersecret, password : String;

const PEAPsuccess, accesslevel, clientfinished, serverfinished, peapkeyingmaterial,uiname,urname : String;

macro ms= prf (pms, mastersecret, (ni1, nr1));

macro finishedi= h1 (ms, clientfinished, (SID, ni1, nr1, csu, {CerR, pk(R)}sk(CA), shd, {pms}pk(R)));

macro finishedr= h1(ms, serverfinished, (SID, ni1, nr1, csu, {CerR, pk(R)}sk(CA), shd, {pms}pk(R), finishedi));

macro ikey= prf ( ms, peapkeyingmaterial, (ni1, nr1));

macro skey= prf ( ms, peapkeyingmaterial, (ni1, nr1));

macro hash= h(n, password);

protocol peap1(I, R, CA)

{

role I

{

const SID, ID, CerR, pms: Data;

fresh ni1: Nonce;

var nr1, n: Nonce;

send\_1 (I, R, SID, ni1, csu);

recv\_4 (R, I, SID, nr1, csu, {CerR, pk(R)}sk(CA), shd);

send\_5 (I, R, {pms}pk(R), ccs);

send\_6 (I,R, finishedi);

recv\_7 (R, I, ccs,finishedr);

match ( h1(ms, serverfinished, (SID, ni1, nr1, csu, {CerR, pk(R)}sk(CA), shd, {pms}pk(R), finishedi)), finishedr);

send\_8 (I,R, {uiname}ikey);

recv\_9 (R, I, {ID, n, uiname}skey);

send\_10 (I, R, {ID, h(n, password), urname}ikey);

recv\_11 (R, I, {PEAPsuccess, accesslevel}skey);

claim\_i1 (I, Secret, CerR);

claim\_i2 (I, Secret, pms);

claim\_i3 (I, Secret, ms);

claim\_i4 (I, Secret, ikey);

claim\_i5 (I, Secret, skey);

claim\_i6 (I, Secret, password);

claim\_i7 (I, Niagree);

claim\_i8 (I, Nisynch);

}

role R

{

const SID, ID, CerR, pms: Data;

fresh nr1, n: Nonce;

var ni1: Nonce;

recv\_1 (I, R, SID, ni1, csu);

send\_2 (R, CA, R);

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

send\_4 (R, I, SID, nr1, csu, {CerR, pk(R)}sk(CA), shd);

recv\_5 (I, R, {pms}pk(R), ccs);

recv\_6 (I,R, finishedi);

match (h1 (ms, clientfinished, (SID, ni1, nr1, csu, {CerR, pk(R)}sk(CA), shd, {pms}pk(R))), finishedi);

send\_7 (R, I, ccs, finishedr);

recv\_8 (I,R, {uiname}ikey);

send\_9 (R, I, {ID, n, uiname}skey);

recv\_10 (I, R, {ID, h(n, password), urname}ikey);

match (hash, h(n,password));

send\_11(R, I, {PEAPsuccess, accesslevel}skey);

claim\_r1 (R, Secret, CerR);

claim\_r2 (R, Secret, pms);

claim\_r3 (R, Secret, ms);

claim\_r4 (R, Secret, ikey);

claim\_r5 (R, Secret, skey);

claim\_r6 (R, Secret, password);

claim\_r7 (R, Niagree);

claim\_r8 (R, Nisynch);

}

role CA

{

const CerR: Data;

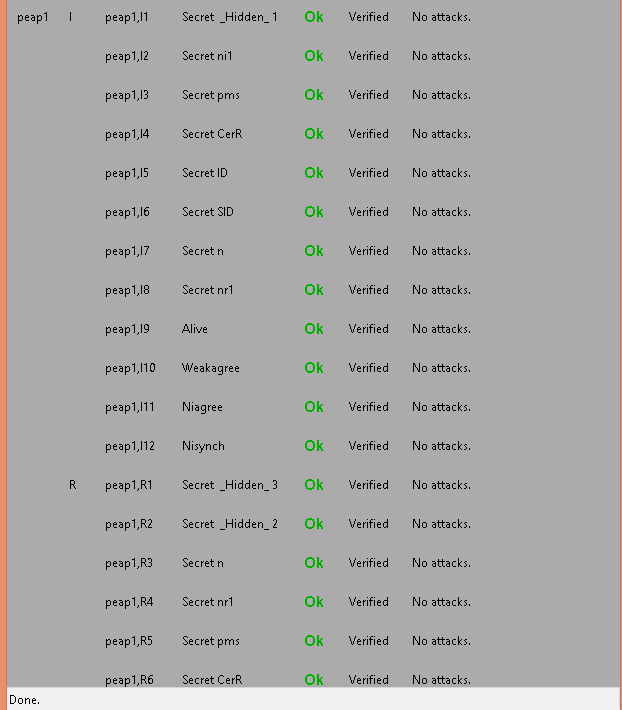
recv\_2(R, CA, R);

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

claim\_ca1 (CA, Secret, CerR);

}

}





MATLAB simulation of PEAP

clc;

clear;

trsae= 0.1667 %rsaencryption

trsad= 0.1667 %rsadecryption

thmacsha256= 0.0714

tsha1= 0.0384; %sha

tsha256= 0.05 %sha256

tmd5= 0.05263; %md5

tdese= 0.29165; %desencryption

tdesd= 0.29165; %desdecryption

%PEAP AUTHENTICATION

peapo\_a = [trsae,5\*thmacsha256+2\*thmacsha256, 2\*thmacsha256,14\*thmacsha256+tdese,tdesd, tmd5+tdese, tdesd];

peapo\_b= [trsad,5\*thmacsha256+2\*thmacsha256, 2\*thmacsha256,14\*thmacsha256+tdesd, tdese,tdesd+tmd5, tdese];

peapo=[trsae,trsad,5\*thmacsha256+2\*thmacsha256, 5\*thmacsha256+ 2\*thmacsha256, 2\*thmacsha256,2\*thmacsha256, 14\*thmacsha256 + tdese,14\*thmacsha256+tdesd, tdese, tdesd, tmd5+tdese,tdesd+tmd5,tdese,tdesd];

%total===============================

for tmp = 2:length(peapo\_a);

peapo\_a (tmp)= peapo\_a (tmp-1)+ peapo\_a (tmp);

end

for tmp = 2:length(peapo);

peapo (tmp)= peapo (tmp-1)+ peapo (tmp);

end

%PEAP AUTHENTICATION Modified

peapm\_a = [trsae, 2\*thmacsha256+tsha1+thmacsha256,thmacsha256, 4\*thmacsha256+6\*tsha1+tdese, tdesd, tmd5+tdese, tdesd];

peapm\_b= [trsad, 2\*thmacsha256+tsha1+thmacsha256,thmacsha256, 4\*thmacsha256+6\*tsha1+tdesd, tdese, tmd5+tdesd, tdese];

peapm=[trsae, trsad, 2\*thmacsha256+tsha1+thmacsha256, 2\*thmacsha256+tsha1+thmacsha256, thmacsha256, thmacsha256, 4\*thmacsha256+6\*tsha1+tdese, 4\*thmacsha256+6\*tsha1+tdesd,tdese, tdesd, tmd5+tdese, tmd5+tdesd, tdese, tdesd];

%total===============================

for tmp = 2:length(peapm\_a);

peapm\_a (tmp)= peapm\_a (tmp-1)+ peapm\_a (tmp);

end

for tmp = 2:length(peapm);

peapm (tmp)= peapm (tmp-1)+ peapm (tmp);

end

total\_number = 100000;

unkown\_attacks= 0;

y\_peapo = zeros(1,10);

y\_peapm= zeros(1,10);

left\_time\_peapo= 0;

left\_time\_peapm=0;

n=1;

for x=0:0.1:0.9

left\_time\_peapo= total\_number\*(1-x)\*peapo\_a(length(peapo\_a));

left\_time\_peapm= total\_number\*(1-x)\*peapm\_a(length(peapm\_a));

unknown\_attacks = uint16(total\_number\*x);

unexpected\_delay\_peapo = randi([1,length(peapo\_a)],1,unknown\_attacks);

unexpected\_delay\_peapm = randi([1,length(peapm\_a)],1,unknown\_attacks);

attack\_total\_delay\_peapo= 0;

attack\_total\_delay\_peapm= 0;

for i=1:unknown\_attacks

attack\_total\_delay\_peapo = attack\_total\_delay\_peapo + peapo\_a(unexpected\_delay\_peapo(i));

attack\_total\_delay\_peapm=attack\_total\_delay\_peapm+peapm\_a(unexpected\_delay\_peapm(i));

end

y\_peapo(n)=(left\_time\_peapo+attack\_total\_delay\_peapo)/(total\_number\*(1-x));

y\_peapm(n)=(left\_time\_peapm+attack\_total\_delay\_peapm)/(total\_number\*(1-x));

n=n+1;

end

x=0:0.1:0.9;

%figure;

plot(x,y\_peapo,'-k', x, y\_peapm,'-.k');

set(gca,'XTick',0:0.1:1);

set(gca, 'xticklabel', {'0','0.1','0.2','0.3','0.4','0.5','0.6','0.7','0.8','0.9','1'});

xlabel('Ratio of unknown to known attacks’,'fontsize',12);

ylabel('Total Time Delay (ms)','fontsize',12);

legend({'PEAPo', 'PEAPm'},'FontSize',12,'FontWeight','bold');

axis([0,0.9,0,10]);

