HATS Deliverable D4.4 Appendix: Automatically generated ABS code for the IDM system

January 31, 2013

Below we show the automatically generated version of the Identity Management system with the lowest security level attached. The code is generated using the module for automated construction of security protocol defined in Task 4.1.

In order to use the module described, the succeeding code should be uploaded into an ABS Eclipse project. This project should also include the two files ABSprotocols and ExecutionEnvironment described in D4.1.

```
module TestProject;

import * from ABSProtocols;
import * from ExecutionEnvironment;



  def Bool
 testmessageFromIDPToAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferIDP
    (ProtocolClause msg)
 = case msg {Message( AgentVar( "IDP" ),   AgentVar( "A" ),

       Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
       Cons ( Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),
       Cons (  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,  Nil
  )  )  )  )     => True ;

     _ => False ;

     }  ;


  def Bool testmessageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonce
IDPNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDP
hashhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDP
nonceANONCEAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOfferIDP
encryptKeyPKIpublicIDPnoncePMSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDP
noncePMSA
(ProtocolClause msg)
 = case msg {Message( AgentVar( "IDP" ),   AgentVar( "A" ),

       Cons (  Encrypt(KeyComposite (
       Cons (  Agent(AgentVar( "IDP" )),
       Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
       Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
       Cons (  Hash(
       Cons ( Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ),
       Cons ( Text("MasterSecret"),
       Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
       Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
  )  )  )  )  ) ,  Nil
  )  )  )  ) ),
       Cons (  Hash(
       Cons (  Hash(
       Cons ( Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ),
       Cons ( Text("MasterSecret"),
       Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
       Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
  )  )  )  )  ) ,
       Cons (  Agent(AgentVar( "A" )),
       Cons (  Agent(AgentVar( "IDP" )),
       Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
       Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
       Cons ( Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),
       Cons (  TextOrigin("CryptoOffer", AgentVar( "A" )) ,
       Cons (  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
       Cons (  Encrypt( KeyPKI(Public,AgentVar( "IDP" )),
       Cons ( Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ),  Nil
  )  ) ,
       Cons (  Encrypt( KeyPKI(Private,AgentVar( "A" )),
       Cons (  Hash(
       Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
```

```
      Cons (  Agent(AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ),  Nil
  )  )  )  ) ,  Nil
  )  ) ,  Nil
  )  )  )  )  )  )  )  )  )  ) ,  Nil
  )  ) ,  Nil
  )  )     => True ;

     _ => False ;

     }  ;


  def Bool testmessageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPN
ONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS
/IDPagentAagentIDPnonceREGISTERAnonceREPLYIDP (ProtocolClause msg)
 = case msg {Message( AgentVar( "IDP" ),   AgentVar( "A" ),

      Cons (  Encrypt(KeyComposite (
      Cons (  Agent(AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ),
      Cons ( Text("MasterSecret"),
      Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
  )  )  )  )  ) ,  Nil
  )  )  )  ) ),
      Cons ( Text("Register to AS/IDP"),
      Cons (  Agent(AgentVar( "A" )),
      Cons (  Agent(AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "REGISTER" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "REPLY" ), AgentVar( "IDP" ) ),  Nil
  )  )  )  )  )  ) ,  Nil
  )  )     => True ;

     _ => False ;

     }  ;


  def Bool testmessageFromIDPToAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferIDP (P
rotocolClause msg)
 = case msg {Message( AgentVar( "IDP" ),   AgentVar( "A" ),

      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons ( Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),
      Cons (  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,  Nil
  )  )  )  )     => True ;

     _ => False ;

     }  ;


  def Bool testmessageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*ID
PNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnon
ce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEA
nonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublic
IDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA
  (ProtocolClause msg)
 = case msg {Message( AgentVar( "IDP" ),   AgentVar( "A" ),

      Cons (  Encrypt(KeyComposite (
      Cons (  Agent(AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),
      Cons ( Text("MasterSecret"),
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
```

```
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
  )  )  )  )  ) ,  Nil
  )  )  )  ) ),
      Cons (  Hash(
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),
      Cons ( Text("MasterSecret"),
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
  )  )  )  )  ) ,
      Cons (  Agent(AgentVar( "A" )),
      Cons (  Agent(AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons ( Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),
      Cons (  TextOrigin("CryptoOffer", AgentVar( "A" )) ,
      Cons (  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
      Cons (  Encrypt( KeyPKI(Public,AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),  Nil
  )  ) ,
      Cons (  Encrypt( KeyPKI(Private,AgentVar( "A" )),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons (  Agent(AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),  Nil
  )  )  )  ) ,  Nil
  )  ) ,  Nil
  )  )  )  )  )  )  )  )  )  ) ,  Nil
  )  ) ,  Nil
  )  )     => True ;

     _ => False ;

     }  ;


  def Bool testmessageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*ID
PNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextRegiste
rtoAS/IDPagentAagentIDPnonceAUTHAnonceAUTHREPLYIDP (ProtocolClause msg)
 = case msg {Message( AgentVar( "IDP" ),   AgentVar( "A" ),

      Cons (  Encrypt(KeyComposite (
      Cons (  Agent(AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),
      Cons ( Text("MasterSecret"),
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
  )  )  )  )  ) ,  Nil
  )  )  )  ) ),
      Cons ( Text("Register to AS/IDP"),
      Cons (  Agent(AgentVar( "A" )),
      Cons (  Agent(AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "AUTH" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ),  Nil
  )  )  )  )  )  ) ,  Nil
  )  )     => True ;

     _ => False ;

     }  ;


  def Bool testmessageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*ID
PNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPnonceAUTHAn
onceAUTHREPLYIDPnonceAUTHREADYSP (ProtocolClause msg)
 = case msg {Message( AgentVar( "IDP" ),   AgentVar( "A" ),

      Cons (  Encrypt(KeyComposite (
      Cons (  Agent(AgentVar( "IDP" )),
```

```
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),
      Cons ( Text("MasterSecret"),
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
 )  )  )  )  ) ,  Nil
 )  )  )  ) ),
      Cons ( Nonce( NonceVar ( "AUTH" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ),
      Cons ( Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),  Nil
 )  )  )  ) ,  Nil
 )  )     => True ;

    _ => False ;

    }  ;


  def Bool testmessageFromSPToAencryptKeyCompositeagentSPnonceAUTHAnonceAUTHREPL
YIDPnonceAUTHREADYSPhashnonceDYNAMICPASSWORDAtextDeliverServicenonceSESSIONIDA (
ProtocolClause msg)
 = case msg {Message( AgentVar( "SP" ),   AgentVar( "A" ),

      Cons (  Encrypt(KeyComposite (
      Cons (  Agent(AgentVar( "SP" )),
      Cons ( Nonce( NonceVar ( "AUTH" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ),
      Cons ( Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) ),  Nil
 )  ) ,  Nil
 )  )  )  )  ) ),
      Cons ( Text("Deliver Service"),
      Cons ( Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) ),  Nil
 )  )  ) ,  Nil
 )  )     => True ;

    _ => False ;

    }  ;



 class  Aclass (AgentTerm name, Network network) implements Agent{
 Bool messageFromIDPToAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferIDPQuery = False
 ;

   Bool messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDP
hashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMa
sterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDP
nonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryp
tKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSAQuery = False  ;

   Bool messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDP
hashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPage
ntAagentIDPnonceREGISTERAnonceREPLYIDPQuery = False  ;

   Bool messageFromIDPToAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferIDPQuery = Fa
lse  ;

   Bool messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEI
DPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSA
textMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*I
DPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonc
e*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSAQuery = False
 ;

   Bool messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEI
DPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextRegistertoAS/I
```

```
DPagentAagentIDPnonceAUTHAnonceAUTHREPLYIDPQuery = False  ;

   Bool messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEI
DPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPnonceAUTHAnonceAUT
HREPLYIDPnonceAUTHREADYSPQuery = False  ;

   Bool messageFromSPToAencryptKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnon
ceAUTHREADYSPhashnonceDYNAMICPASSWORDAtextDeliverServicenonceSESSIONIDAQuery = F
alse  ;

    PayloadElement agentA =  UndefinedElem   ;

    PayloadElement newnonceANONCEA =  UndefinedElem   ;

    PayloadElement newnonceSIDA =  UndefinedElem   ;

    PayloadElement textCryptoOfferA =  UndefinedElem   ;

    PayloadElement agentIDP =  UndefinedElem   ;

    ProtocolClause messageFromAToIDPagentAnonceANONCEAnonceSIDAtextCryptoOfferA
=  UndefinedClause  ;

    ProtocolClause messageFromIDPToAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferIDP
=  UndefinedClause   ;

    PayloadElement nonceIDPNONCEIDP =  UndefinedElem   ;

    PayloadElement textCryptoOfferIDP =  UndefinedElem   ;

    PayloadElement newnoncePMSA =  UndefinedElem   ;

    PayloadElement newkeyKeyPKIprivateIDP =  UndefinedElem   ;

    PayloadElement newkeyKeyPKIpublicIDP =  UndefinedElem   ;

    PayloadElement encryptKeyPKIpublicIDPnoncePMSA =  UndefinedElem   ;

    ProtocolClause messageFromAToIDPencryptKeyPKIpublicIDPnoncePMSA =  Undefined
Clause  ;

    PayloadElement hashnonceIDPNONCEIDPagentIDPnoncePMSA =  UndefinedElem   ;


    PayloadElement newkeyKeyPKIprivateA =  UndefinedElem   ;

    PayloadElement newkeyKeyPKIpublicA =  UndefinedElem   ;

    PayloadElement encryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =
 UndefinedElem   ;

    ProtocolClause messageFromAToIDPencryptKeyPKIprivateAhashnonceIDPNONCEIDPage
ntIDPnoncePMSA =  UndefinedClause  ;

    PayloadElement textMasterSecret =  UndefinedElem   ;

    PayloadElement hashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDP =
UndefinedElem   ;

    PayloadElement hashhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDP
agentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOff
erIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPage
ntIDPnoncePMSA =  UndefinedElem   ;

    PayloadElement keyKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnoncePMS
AtextMasterSecretnonceANONCEAnonceIDPNONCEIDP =  UndefinedElem   ;

    PayloadElement encryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnonc
ePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMasterSecr
etnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonceSID
AtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyPKIp
rivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =  UndefinedElem   ;
```

```
    ProtocolClause messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceID
PNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnonceP
MSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceID
PNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonceP
MSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =  UndefinedClause
  ;

    ProtocolClause messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonce
IDPNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnonc
ePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonce
IDPNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonc
ePMSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =  UndefinedClau
se   ;

    PayloadElement keyKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashnonceP
MSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDP =  UndefinedElem   ;

    PayloadElement decryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashno
ncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPencryptKeyCompositeagentIDPno
nceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCE
IDPhashhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPno
nceANONCEAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyP
KIpublicIDPnoncePMSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =
  UndefinedElem   ;

    PayloadElement hashhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDP
agentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOff
erIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPage
ntIDPnoncePMSA =  UndefinedElem   ;

    PayloadElement textRegistertoAS/IDP =  UndefinedElem   ;

    PayloadElement newnonceDYNAMICPASSWORDA =  UndefinedElem   ;

    PayloadElement newnonceREGISTERA =  UndefinedElem   ;

    PayloadElement encryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnonc
ePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPnonceDYNAMI
CPASSWORDAagentAnonceREGISTERA =  UndefinedElem   ;

    ProtocolClause messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceID
PNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegisterto
AS/IDPnonceDYNAMICPASSWORDAagentAnonceREGISTERA =  UndefinedClause  ;

    ProtocolClause messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonce
IDPNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegister
toAS/IDPagentAagentIDPnonceREGISTERAnonceREPLYIDP =  UndefinedClause   ;

    PayloadElement decryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashno
ncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPencryptKeyCompositeagentIDPno
nceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCE
IDPtextRegistertoAS/IDPagentAagentIDPnonceREGISTERAnonceREPLYIDP =  UndefinedEle
m   ;

    PayloadElement nonceREPLYIDP =  UndefinedElem   ;

    PayloadElement newnonce*ANONCEA =  UndefinedElem   ;

    PayloadElement newnonce*SIDA =  UndefinedElem   ;

    ProtocolClause messageFromAToIDPagentAnonce*ANONCEAnonce*SIDAtextCryptoOffer
A =  UndefinedClause  ;

    ProtocolClause messageFromIDPToAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferID
P =  UndefinedClause   ;

    PayloadElement nonce*IDPNONCEIDP =  UndefinedElem   ;

    PayloadElement newnonce*PMSA =  UndefinedElem   ;

    PayloadElement encryptKeyPKIpublicIDPnonce*PMSA =  UndefinedElem   ;
```

```
    ProtocolClause messageFromAToIDPencryptKeyPKIpublicIDPnonce*PMSA =  Undefine
dClause  ;

    PayloadElement hashnonce*IDPNONCEIDPagentIDPnonce*PMSA =  UndefinedElem   ;


    PayloadElement encryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA
=  UndefinedElem   ;

    ProtocolClause messageFromAToIDPencryptKeyPKIprivateAhashnonce*IDPNONCEIDPag
entIDPnonce*PMSA =  UndefinedClause  ;

    PayloadElement hashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDP
=  UndefinedElem   ;

    PayloadElement hashhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCE
IDPagentAagentIDPnonce*ANONCEAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCry
ptoOfferIDPencryptKeyPKIpublicIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNON
CEIDPagentIDPnonce*PMSA =  UndefinedElem   ;

    PayloadElement keyKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*
PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDP =  UndefinedElem   ;

    PayloadElement encryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashno
nce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSAtextMast
erSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*IDPNONCEI
DPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonce*PMSAen
cryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA =  UndefinedElem   ;


    ProtocolClause messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*
IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashn
once*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONC
EAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpubl
icIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA =  U
ndefinedClause  ;

    ProtocolClause messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonc
e*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhas
hnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANO
NCEAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpu
blicIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA =
 UndefinedClause   ;

    PayloadElement keyKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonc
e*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDP =  UndefinedElem   ;

    PayloadElement decryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhash
nonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPencryptKeyCompositeagent
IDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonc
e*IDPNONCEIDPhashhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPage
ntAagentIDPnonce*ANONCEAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOff
erIDPencryptKeyPKIpublicIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPa
gentIDPnonce*PMSA =  UndefinedElem   ;

    PayloadElement hashhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCE
IDPagentAagentIDPnonce*ANONCEAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCry
ptoOfferIDPencryptKeyPKIpublicIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNON
CEIDPagentIDPnonce*PMSA =  UndefinedElem   ;

    PayloadElement textauthenticatetoservice =  UndefinedElem   ;

    PayloadElement agentSP =  UndefinedElem   ;

    PayloadElement newnonceAUTHA =  UndefinedElem   ;

    PayloadElement encryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashno
nce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatetoservicea
gentAagentSPnonceAUTHA =  UndefinedElem   ;

    ProtocolClause messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*
```

```
IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthe
nticatetoserviceagentAagentSPnonceAUTHA =  UndefinedClause  ;

    ProtocolClause messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonc
e*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextReg
istertoAS/IDPagentAagentIDPnonceAUTHAnonceAUTHREPLYIDP =  UndefinedClause   ;


    PayloadElement decryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhash
nonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPencryptKeyCompositeagent
IDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonc
e*IDPNONCEIDPtextRegistertoAS/IDPagentAagentIDPnonceAUTHAnonceAUTHREPLYIDP =  Un
definedElem   ;

    PayloadElement nonceAUTHREPLYIDP =  UndefinedElem   ;

    PayloadElement encryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashno
nce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatetoservicea
gentAnonceDYNAMICPASSWORDAnonceAUTHAnonceAUTHREPLYIDP =  UndefinedElem   ;

    ProtocolClause messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*
IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthe
nticatetoserviceagentAnonceDYNAMICPASSWORDAnonceAUTHAnonceAUTHREPLYIDP =  Undefi
nedClause  ;

    ProtocolClause messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonc
e*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPnonceAU
THAnonceAUTHREPLYIDPnonceAUTHREADYSP =  UndefinedClause   ;

    PayloadElement decryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhash
nonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPencryptKeyCompositeagent
IDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonc
e*IDPNONCEIDPnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREADYSP =  UndefinedElem   ;


    PayloadElement nonceAUTHREADYSP =  UndefinedElem   ;

    PayloadElement textRequestService =  UndefinedElem   ;

    PayloadElement newnonceSESSIONIDA =  UndefinedElem   ;

    PayloadElement hashnonceDYNAMICPASSWORDA =  UndefinedElem   ;

    PayloadElement keyKeyCompositeagentAnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREAD
YSPhashnonceDYNAMICPASSWORDA =  UndefinedElem   ;

    PayloadElement encryptKeyCompositeagentAnonceAUTHAnonceAUTHREPLYIDPnonceAUTH
READYSPhashnonceDYNAMICPASSWORDAtextRequestServicenonceSESSIONIDA =  UndefinedEl
em   ;

    ProtocolClause messageFromAToSPencryptKeyCompositeagentAnonceAUTHAnonceAUTHR
EPLYIDPnonceAUTHREADYSPhashnonceDYNAMICPASSWORDAtextRequestServicenonceSESSIONID
A =  UndefinedClause  ;

    ProtocolClause messageFromSPToAencryptKeyCompositeagentSPnonceAUTHAnonceAUTH
REPLYIDPnonceAUTHREADYSPhashnonceDYNAMICPASSWORDAtextDeliverServicenonceSESSIONI
DA =  UndefinedClause   ;

    PayloadElement keyKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREA
DYSPhashnonceDYNAMICPASSWORDA =  UndefinedElem   ;

    PayloadElement decryptKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnonceAUT
HREADYSPhashnonceDYNAMICPASSWORDAencryptKeyCompositeagentSPnonceAUTHAnonceAUTHRE
PLYIDPnonceAUTHREADYSPhashnonceDYNAMICPASSWORDAtextDeliverServicenonceSESSIONIDA
 =  UndefinedElem   ;

    PayloadElement textDeliverService =  UndefinedElem   ;

      Unit run(){ await(network != null) ;

     agentA =  Agent(AgentVar( "A" )) ;
```

```
   newnonceANONCEA =  New(Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ) ) ;


   newnonceSIDA =  New(Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ) ) ;

   textCryptoOfferA =  TextOrigin("CryptoOffer", AgentVar( "A" ))  ;

   agentIDP =  Agent(AgentVar( "IDP" )) ;

   messageFromAToIDPagentAnonceANONCEAnonceSIDAtextCryptoOfferA = Message(AgentV
ar( "A" ),AgentVar( "IDP" ),
  list[  Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) )
, Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("CryptoOffer", Agent
Var( "A" )) ]) ;

   network!send( Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[  Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) )
, Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("CryptoOffer", Agent
Var( "A" )) ]) )  ;

   await( messageFromIDPToAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferIDPQuery )  ;


   messageFromIDPToAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferIDP = Message(AgentV
ar( "IDP" ),AgentVar( "A" ),
  list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "
SID" ), AgentVar( "A" ) ),  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ]) ;


   nonceIDPNONCEIDP = Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ) ;

   textCryptoOfferIDP =  TextOrigin("CryptoOffer", AgentVar( "IDP" ))  ;

   newnoncePMSA =  New(Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ) ) ;

   newkeyKeyPKIprivateIDP =  New(Key( KeyPKI(Private,AgentVar( "IDP" )))) ;

   newkeyKeyPKIpublicIDP =  New(Key( KeyPKI(Public,AgentVar( "IDP" )))) ;

   encryptKeyPKIpublicIDPnoncePMSA =
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ) ;

   messageFromAToIDPencryptKeyPKIpublicIDPnoncePMSA = Message(AgentVar( "A" ),Ag
entVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] )]) ;

   network!send( Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] )]) )  ;

   hashnonceIDPNONCEIDPagentIDPnoncePMSA =
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] ) ;

   newkeyKeyPKIprivateA =  New(Key( KeyPKI(Private,AgentVar( "A" )))) ;

   newkeyKeyPKIpublicA =  New(Key( KeyPKI(Public,AgentVar( "A" )))) ;

   encryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] ) ;

   messageFromAToIDPencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =
 Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
```

```
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )]) ;

   network!send( Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )]) )  ;

   textMasterSecret = Text("MasterSecret") ;

   hashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDP =
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ) ;

   hashhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPno
nceANONCEAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyP
KIpublicIDPnoncePMSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =

 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] ) ;

   keyKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMasterSecre
tnonceANONCEAnonceIDPNONCEIDP =  Key( KeyComposite(list[   Agent(AgentVar( "A" )
), Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE"
 ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) ) ;

   encryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMasterS
ecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMasterSecretnonceANONCEAno
nceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonceSIDAtextCryptoOffer
AtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyPKIprivateAhashnonce
IDPNONCEIDPagentIDPnoncePMSA =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONC
E" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )] ) ;

   messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnon
cePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMasterSec
retnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonceSI
DAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyPKI
privateAhashnonceIDPNONCEIDPagentIDPnoncePMSA = Message(AgentVar( "A" ),AgentVar
( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONC
E" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
```

```
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )] )] )]) ;

   network!send( Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONC
E" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )] )] )]) )  ;

   await( messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEI
DPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtext
MasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEI
DPnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencr
yptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSAQuery )  ;

   messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashn
oncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMasterS
ecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonce
SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyP
KIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA = Message(AgentVar( "IDP" ),Agen
tVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "ANO
NCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )] )] )]) ;

   keyKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMasterSec
retnonceANONCEAnonceIDPNONCEIDP =  Key( KeyComposite(list[   Agent(AgentVar( "ID
P" )), Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNO
NCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
```

```
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) ) ;

   decryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMaste
rSecretnonceANONCEAnonceIDPNONCEIDPencryptKeyCompositeagentIDPnonceANONCEAnonceI
DPNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnonce
PMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceI
DPNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonce
PMSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =
  Decrypt(   KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "A
NONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "ANO
NCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )] ) ] ) ;

   hashhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPno
nceANONCEAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyP
KIpublicIDPnoncePMSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =

 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] ) ;

   textRegistertoAS/IDP = Text("Register to AS/IDP") ;

   newnonceDYNAMICPASSWORDA =  New(Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentV
ar( "A" ) ) ) ;

   newnonceREGISTERA =  New(Nonce( NonceVar ( "REGISTER" ), AgentVar( "A" ) ) )
;

   encryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMasterS
ecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPnonceDYNAMICPASSWORDAagentA
nonceREGISTERA =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONC
E" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"), Nonce( NonceVar
( "DYNAMICPASSWORD" ), AgentVar( "A" ) ),  Agent(AgentVar( "A" )), Nonce( NonceV
ar ( "REGISTER" ), AgentVar( "A" ) )] ) ;

   messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnon
cePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPnonceDYNAM
ICPASSWORDAagentAnonceREGISTERA = Message(AgentVar( "A" ),AgentVar( "IDP" ),
```

```
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONC
E" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"), Nonce( NonceVar
( "DYNAMICPASSWORD" ), AgentVar( "A" ) ),  Agent(AgentVar( "A" )), Nonce( NonceV
ar ( "REGISTER" ), AgentVar( "A" ) )] )]) ;

   network!send( Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONC
E" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"), Nonce( NonceVar
( "DYNAMICPASSWORD" ), AgentVar( "A" ) ),  Agent(AgentVar( "A" )), Nonce( NonceV
ar ( "REGISTER" ), AgentVar( "A" ) )] )]) )  ;

   await( messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEI
DPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPa
gentAagentIDPnonceREGISTERAnonceREPLYIDPQuery )  ;

   messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashn
oncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPagentAag
entIDPnonceREGISTERAnonceREPLYIDP = Message(AgentVar( "IDP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "ANO
NCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"),  Agent(AgentVar(
 "A" )),  Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "REGISTER" ), AgentVar( "A
" ) ), Nonce( NonceVar ( "REPLY" ), AgentVar( "IDP" ) )] )]) ;

   decryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMaste
rSecretnonceANONCEAnonceIDPNONCEIDPencryptKeyCompositeagentIDPnonceANONCEAnonceI
DPNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistert
oAS/IDPagentAagentIDPnonceREGISTERAnonceREPLYIDP =
  Decrypt(   KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "A
NONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "ANO
NCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"),  Agent(AgentVar(
 "A" )),  Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "REGISTER" ), AgentVar( "A
" ) ), Nonce( NonceVar ( "REPLY" ), AgentVar( "IDP" ) )] ) ] ) ;

   nonceREPLYIDP = Nonce( NonceVar ( "REPLY" ), AgentVar( "IDP" ) ) ;

   newnonce*ANONCEA =  New(Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ) ) ;


   newnonce*SIDA =  New(Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ) ) ;

   messageFromAToIDPagentAnonce*ANONCEAnonce*SIDAtextCryptoOfferA = Message(Agen
tVar( "A" ),AgentVar( "IDP" ),
  list[  Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" )
), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOrigin("CryptoOffer", Age
ntVar( "A" )) ]) ;

   network!send( Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[  Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" )
), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOrigin("CryptoOffer", Age
ntVar( "A" )) ]) )  ;

   await( messageFromIDPToAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferIDPQuery )
```

```
 ;

   messageFromIDPToAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferIDP = Message(Agen
tVar( "IDP" ),AgentVar( "A" ),
  list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ), Nonce( NonceVar (
"*SID" ), AgentVar( "A" ) ),  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ]) ;


   nonce*IDPNONCEIDP = Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ) ;


   newnonce*PMSA =  New(Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ) ) ;

   encryptKeyPKIpublicIDPnonce*PMSA =
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ) ;

   messageFromAToIDPencryptKeyPKIpublicIDPnonce*PMSA = Message(AgentVar( "A" ),A
gentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] )]) ;

   network!send( Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] )]) )  ;

   hashnonce*IDPNONCEIDPagentIDPnonce*PMSA =
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] ) ;

   encryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA =
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] ) ;

   messageFromAToIDPencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA
 = Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )]) ;

   network!send( Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )]) )  ;

   hashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDP =
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ) ;

   hashhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentID
Pnonce*ANONCEAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencry
ptKeyPKIpublicIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnon
ce*PMSA =
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] ) ;
```

```
   keyKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMasterSe
cretnonce*ANONCEAnonce*IDPNONCEIDP =  Key( KeyComposite(list[   Agent(AgentVar(
"A" )), Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*ID
PNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) ) ;

   encryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMast
erSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSAtextMasterSecretnonce*AN
ONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*IDPNONCEIDPnonce*SIDAtext
CryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonce*PMSAencryptKeyPKIpriva
teAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )] ) ;

   messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashn
once*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSAtextMas
terSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*IDPNONCE
IDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonce*PMSAe
ncryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA = Message(AgentVar(
"A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )]) ;

   network!send( Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
```

```
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )] )] )]) )  ;


   await( messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONC
EIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PM
SAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce
*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPno
nce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSAQuery )  ;


   messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhas
hnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSAtextM
asterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*IDPNON
CEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonce*PMS
AencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA = Message(AgentVar
( "IDP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )] )] )]) ;

   keyKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMaster
Secretnonce*ANONCEAnonce*IDPNONCEIDP =  Key( KeyComposite(list[   Agent(AgentVar
( "IDP" )), Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar (
"*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) ) ;

   decryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMa
sterSecretnonce*ANONCEAnonce*IDPNONCEIDPencryptKeyCompositeagentIDPnonce*ANONCEA
nonce*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhas
hhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce
*ANONCEAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyP
KIpublicIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMS
A =
  Decrypt(   KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*
ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" )
 ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
```

```
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )] ) ] ) ;


   hashhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentID
Pnonce*ANONCEAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencry
ptKeyPKIpublicIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnon
ce*PMSA =
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )] ) ;

   textauthenticatetoservice = Text("authenticate to service") ;

   agentSP =  Agent(AgentVar( "SP" )) ;

   newnonceAUTHA =  New(Nonce( NonceVar ( "AUTH" ), AgentVar( "A" ) ) ) ;

   encryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMast
erSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatetoserviceagentAagentSPnonc
eAUTHA =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("authenticate to service"),  Agent(A
gentVar( "A" )),  Agent(AgentVar( "SP" )), Nonce( NonceVar ( "AUTH" ), AgentVar(
 "A" ) )] ) ;

   messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashn
once*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatetoservice
agentAagentSPnonceAUTHA = Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("authenticate to service"),  Agent(A
gentVar( "A" )),  Agent(AgentVar( "SP" )), Nonce( NonceVar ( "AUTH" ), AgentVar(
 "A" ) )] )]) ;

   network!send( Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("authenticate to service"),  Agent(A
gentVar( "A" )),  Agent(AgentVar( "SP" )), Nonce( NonceVar ( "AUTH" ), AgentVar(
 "A" ) )] )]) )  ;

   await( messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONC
EIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextRegistertoAS
```

```
/IDPagentAagentIDPnonceAUTHAnonceAUTHREPLYIDPQuery )  ;

   messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhas
hnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextRegistertoAS/IDPage
ntAagentIDPnonceAUTHAnonceAUTHREPLYIDP = Message(AgentVar( "IDP" ),AgentVar( "A"
 ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"),  Agent(AgentV
ar( "A" )),  Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "AUTH" ), AgentVar( "A"
 ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) )] )]) ;

   decryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMa
sterSecretnonce*ANONCEAnonce*IDPNONCEIDPencryptKeyCompositeagentIDPnonce*ANONCEA
nonce*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtex
tRegistertoAS/IDPagentAagentIDPnonceAUTHAnonceAUTHREPLYIDP =
  Decrypt(   KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*
ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" )
 ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"),  Agent(AgentV
ar( "A" )),  Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "AUTH" ), AgentVar( "A"
 ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) )] ) ] ) ;

   nonceAUTHREPLYIDP = Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ) ;


   encryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMast
erSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatetoserviceagentAnonceDYNAMI
CPASSWORDAnonceAUTHAnonceAUTHREPLYIDP =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("authenticate to service"),  Agent(A
gentVar( "A" )), Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) ), Nonce
( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agent
Var( "IDP" ) )] ) ;

   messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashn
once*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatetoservice
agentAnonceDYNAMICPASSWORDAnonceAUTHAnonceAUTHREPLYIDP = Message(AgentVar( "A" )
,AgentVar( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("authenticate to service"),  Agent(A
gentVar( "A" )), Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) ), Nonce
( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agent
Var( "IDP" ) )] )]) ;

   network!send( Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("authenticate to service"),  Agent(A
gentVar( "A" )), Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) ), Nonce
```

```
( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agent
Var( "IDP" ) )] )]) )  ;

   await( messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONC
EIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPnonceAUTHAnonceA
UTHREPLYIDPnonceAUTHREADYSPQuery )  ;

   messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhas
hnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPnonceAUTHAnonceAUTHREPL
YIDPnonceAUTHREADYSP = Message(AgentVar( "IDP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Nonce( NonceVar ( "AUTH" ), AgentVar( "A"
 ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "A
UTHREADY" ), AgentVar( "SP" ) )] )]) ;

   decryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMa
sterSecretnonce*ANONCEAnonce*IDPNONCEIDPencryptKeyCompositeagentIDPnonce*ANONCEA
nonce*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPnon
ceAUTHAnonceAUTHREPLYIDPnonceAUTHREADYSP =
  Decrypt(   KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*
ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" )
 ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Nonce( NonceVar ( "AUTH" ), AgentVar( "A"
 ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "A
UTHREADY" ), AgentVar( "SP" ) )] ) ] ) ;

   nonceAUTHREADYSP = Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ) ;

   textRequestService = Text("Request Service") ;

   newnonceSESSIONIDA =  New(Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) )
) ;

   hashnonceDYNAMICPASSWORDA =
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] ) ;

   keyKeyCompositeagentAnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREADYSPhashnonceDYNA
MICPASSWORDA =  Key( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVa
r ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP
" ) ), Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) ) ;


   encryptKeyCompositeagentAnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREADYSPhashnonce
DYNAMICPASSWORDAtextRequestServicenonceSESSIONIDA =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "AUTH"
 ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), Non
ce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) , li
st[ Text("Request Service"), Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) )]
 ) ;

   messageFromAToSPencryptKeyCompositeagentAnonceAUTHAnonceAUTHREPLYIDPnonceAUTH
READYSPhashnonceDYNAMICPASSWORDAtextRequestServicenonceSESSIONIDA = Message(Agen
tVar( "A" ),AgentVar( "SP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "AUTH"
 ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), Non
ce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
```

```
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) , li
st[ Text("Request Service"), Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) )]
 )]) ;

   network!send( Message(AgentVar( "A" ),AgentVar( "SP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "AUTH"
 ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), Non
ce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) , li
st[ Text("Request Service"), Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) )]
 )]) )  ;

   await( messageFromSPToAencryptKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPn
onceAUTHREADYSPhashnonceDYNAMICPASSWORDAtextDeliverServicenonceSESSIONIDAQuery )
  ;

   messageFromSPToAencryptKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnonceAUT
HREADYSPhashnonceDYNAMICPASSWORDAtextDeliverServicenonceSESSIONIDA = Message(Age
ntVar( "SP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "SP" )), Nonce( NonceVar ( "AUTH
" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), No
nce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) , li
st[ Text("Deliver Service"), Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) )]
 )]) ;

   keyKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREADYSPhashnonceDYN
AMICPASSWORDA =  Key( KeyComposite(list[   Agent(AgentVar( "SP" )), Nonce( Nonce
Var ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "I
DP" ) ), Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) ) ;


   decryptKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREADYSPhashnonc
eDYNAMICPASSWORDAencryptKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnonceAUTHR
EADYSPhashnonceDYNAMICPASSWORDAtextDeliverServicenonceSESSIONIDA =
  Decrypt(   KeyComposite(list[   Agent(AgentVar( "SP" )), Nonce( NonceVar ( "AU
TH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ),
Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) , li
st[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "SP" )), Nonce( NonceVar ( "AUTH
" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), No
nce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) , li
st[ Text("Deliver Service"), Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) )]
 ) ] ) ;

   textDeliverService = Text("Deliver Service") ;

     }
   /* End of run-method */


    Unit receive(ProtocolClause msg){
  if ( testmessageFromIDPToAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferIDP (msg) )
  {messageFromIDPToAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferIDPQuery = True ;


     }
     else  {
  if ( testmessageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCE
IDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtex
tMasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCE
IDPnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAenc
ryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA (msg) )
  {messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashn
oncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMasterS
ecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonce
SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyP
```

```
KIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSAQuery = True ;

     }
     else  {
  if ( testmessageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCE
IDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDP
agentAagentIDPnonceREGISTERAnonceREPLYIDP (msg) )
  {messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashn
oncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPagentAag
entIDPnonceREGISTERAnonceREPLYIDPQuery = True ;

     }
     else  {
  if ( testmessageFromIDPToAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferIDP (msg)
)
  {messageFromIDPToAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferIDPQuery = True ;


     }
     else  {
  if ( testmessageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNON
CEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*P
MSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonc
e*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPn
once*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA (msg) )
  {messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhas
hnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSAtextM
asterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*IDPNON
CEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonce*PMS
AencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSAQuery = True ;

     }
     else  {
  if ( testmessageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNON
CEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextRegistertoA
S/IDPagentAagentIDPnonceAUTHAnonceAUTHREPLYIDP (msg) )
  {messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhas
hnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextRegistertoAS/IDPage
ntAagentIDPnonceAUTHAnonceAUTHREPLYIDPQuery = True ;

     }
     else  {
  if ( testmessageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNON
CEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPnonceAUTHAnonce
AUTHREPLYIDPnonceAUTHREADYSP (msg) )
  {messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhas
hnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPnonceAUTHAnonceAUTHREPL
YIDPnonceAUTHREADYSPQuery = True ;

     }
     else  {
  if ( testmessageFromSPToAencryptKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDP
nonceAUTHREADYSPhashnonceDYNAMICPASSWORDAtextDeliverServicenonceSESSIONIDA (msg)
 )
  {messageFromSPToAencryptKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnonceAUT
HREADYSPhashnonceDYNAMICPASSWORDAtextDeliverServicenonceSESSIONIDAQuery = True ;


     }
     else  { skip  ;

       }
   /* End of test method */

     }
   /* End of test method */

     }
   /* End of test method */

     }
   /* End of test method */
```

```
    }
   /* End of test method */

    }
   /* End of test method */

    }
   /* End of test method */

    }
   /* End of test method */

  }
   /* End of receive method */

  }
  /* End of class */

  def Bool testmessageFromAToIDPagentAnonceANONCEAnonceSIDAtextCryptoOfferA (Pro
tocolClause msg)
 = case msg {Message( AgentVar( "A" ),   AgentVar( "IDP" ),

       Cons (  Agent(AgentVar( "A" )),
       Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
       Cons ( Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),
       Cons (  TextOrigin("CryptoOffer", AgentVar( "A" )) ,  Nil
  )  )  )  )  )     => True ;

    _ => False ;

    }  ;


  def Bool testmessageFromAToIDPencryptKeyPKIpublicIDPnoncePMSA (ProtocolClause
msg)
 = case msg {Message( AgentVar( "A" ),   AgentVar( "IDP" ),

       Cons (  Encrypt( KeyPKI(Public,AgentVar( "IDP" )),
       Cons ( Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ),  Nil
  )  ) ,  Nil
  )  )     => True ;

    _ => False ;

    }  ;


  def Bool testmessageFromAToIDPencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentID
PnoncePMSA (ProtocolClause msg)
 = case msg {Message( AgentVar( "A" ),   AgentVar( "IDP" ),

       Cons (  Encrypt( KeyPKI(Private,AgentVar( "A" )),
       Cons (  Hash(
       Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
       Cons (  Agent(AgentVar( "IDP" )),
       Cons ( Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ),  Nil
  )  )  )  ) ,  Nil
  )  ) ,  Nil
  )  )     => True ;

    _ => False ;

    }  ;


  def Bool testmessageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNON
CEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAt
extMasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNON
CEIDPnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAe
ncryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA (ProtocolClause msg)
 = case msg {Message( AgentVar( "A" ),   AgentVar( "IDP" ),
```

```
      Cons (  Encrypt(KeyComposite (
      Cons (  Agent(AgentVar( "A" )),
      Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ),
      Cons ( Text("MasterSecret"),
      Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
 )  )  )  )  ) ,  Nil
 )  )  )  ) ),
      Cons (  Hash(
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ),
      Cons ( Text("MasterSecret"),
      Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
 )  )  )  )  ) ,
      Cons (  Agent(AgentVar( "A" )),
      Cons (  Agent(AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons ( Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),
      Cons (  TextOrigin("CryptoOffer", AgentVar( "A" )) ,
      Cons (  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
      Cons (  Encrypt( KeyPKI(Public,AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ),  Nil
 )  ) ,
      Cons (  Encrypt( KeyPKI(Private,AgentVar( "A" )),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons (  Agent(AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ),  Nil
 )  )  )  ) ,  Nil
 )  ) ,  Nil
 )  )  )  )  )  )  )  )  )  ) ,  Nil
 )  ) ,  Nil
 )  )     => True ;

    _ => False ;

    }  ;


  def Bool testmessageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNON
CEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/I
DPnonceDYNAMICPASSWORDAagentAnonceREGISTERA (ProtocolClause msg)
 = case msg {Message( AgentVar( "A" ),   AgentVar( "IDP" ),

      Cons (  Encrypt(KeyComposite (
      Cons (  Agent(AgentVar( "A" )),
      Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ),
      Cons ( Text("MasterSecret"),
      Cons ( Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
 )  )  )  )  ) ,  Nil
 )  )  )  ) ),
      Cons ( Text("Register to AS/IDP"),
      Cons ( Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) ),
      Cons (  Agent(AgentVar( "A" )),
      Cons ( Nonce( NonceVar ( "REGISTER" ), AgentVar( "A" ) ),  Nil
 )  )  )  )  ) ,  Nil
 )  )     => True ;

    _ => False ;

    }  ;
```

```
  def Bool testmessageFromAToIDPagentAnonce*ANONCEAnonce*SIDAtextCryptoOfferA (P
rotocolClause msg)
 = case msg {Message( AgentVar( "A" ),   AgentVar( "IDP" ),

       Cons (  Agent(AgentVar( "A" )),
       Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
       Cons ( Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),
       Cons (  TextOrigin("CryptoOffer", AgentVar( "A" )) ,  Nil
  )  )  )  )  )     => True ;

     _ => False ;

     }  ;


  def Bool testmessageFromAToIDPencryptKeyPKIpublicIDPnonce*PMSA (ProtocolClause
 msg)
 = case msg {Message( AgentVar( "A" ),   AgentVar( "IDP" ),

       Cons (  Encrypt( KeyPKI(Public,AgentVar( "IDP" )),
       Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),  Nil
  )  ) ,  Nil
  )  )     => True ;

     _ => False ;

     }  ;


  def Bool testmessageFromAToIDPencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentI
DPnonce*PMSA (ProtocolClause msg)
 = case msg {Message( AgentVar( "A" ),   AgentVar( "IDP" ),

       Cons (  Encrypt( KeyPKI(Private,AgentVar( "A" )),
       Cons (  Hash(
       Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
       Cons (  Agent(AgentVar( "IDP" )),
       Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),  Nil
  )  )  )  ) ,  Nil
  )  ) ,  Nil
  )  )     => True ;

     _ => False ;

     }  ;


  def Bool testmessageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPN
ONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce
*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAno
nce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicID
Pnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA (Protoco
lClause msg)
 = case msg {Message( AgentVar( "A" ),   AgentVar( "IDP" ),

       Cons (  Encrypt(KeyComposite (
       Cons (  Agent(AgentVar( "A" )),
       Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
       Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
       Cons (  Hash(
       Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),
       Cons ( Text("MasterSecret"),
       Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
       Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
  )  )  )  )  ) ,  Nil
  )  )  )  ) ),
       Cons (  Hash(
       Cons (  Hash(
       Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),
       Cons ( Text("MasterSecret"),
       Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
```

```
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
 )  )  )  )  ) ,
      Cons (  Agent(AgentVar( "A" )),
      Cons (  Agent(AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons ( Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),
      Cons (  TextOrigin("CryptoOffer", AgentVar( "A" )) ,
      Cons (  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
      Cons (  Encrypt( KeyPKI(Public,AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),  Nil
 )  ) ,
      Cons (  Encrypt( KeyPKI(Private,AgentVar( "A" )),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons (  Agent(AgentVar( "IDP" )),
      Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),  Nil
 )  )  )  ) ,  Nil
 )  ) ,  Nil
 )  )  )  )  )  )  )  )  )  ) ,  Nil
 )  ) ,  Nil
 )  )     => True ;

    _ => False ;

    }  ;


  def Bool testmessageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPN
ONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthentic
atetoserviceagentAagentSPnonceAUTHA (ProtocolClause msg)
 = case msg {Message( AgentVar( "A" ),   AgentVar( "IDP" ),

      Cons (  Encrypt(KeyComposite (
      Cons (  Agent(AgentVar( "A" )),
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),
      Cons ( Text("MasterSecret"),
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
 )  )  )  )  ) ,  Nil
 )  )  )  ) ),
      Cons ( Text("authenticate to service"),
      Cons (  Agent(AgentVar( "A" )),
      Cons (  Agent(AgentVar( "SP" )),
      Cons ( Nonce( NonceVar ( "AUTH" ), AgentVar( "A" ) ),  Nil
 )  )  )  )  ) ,  Nil
 )  )     => True ;

    _ => False ;

    }  ;


  def Bool testmessageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPN
ONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthentic
atetoserviceagentAnonceDYNAMICPASSWORDAnonceAUTHAnonceAUTHREPLYIDP (ProtocolClau
se msg)
 = case msg {Message( AgentVar( "A" ),   AgentVar( "IDP" ),

      Cons (  Encrypt(KeyComposite (
      Cons (  Agent(AgentVar( "A" )),
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ),
      Cons ( Text("MasterSecret"),
      Cons ( Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Nil
 )  )  )  )  ) ,  Nil
```

```
  )  )  )  ) ),
       Cons ( Text("authenticate to service"),
       Cons (  Agent(AgentVar( "A" )),
       Cons ( Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) ),
       Cons ( Nonce( NonceVar ( "AUTH" ), AgentVar( "A" ) ),
       Cons ( Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ),  Nil
  )  )  )  )  )  ) ,  Nil
  )  )     => True ;

     _ => False ;

     }  ;


  def Bool testmessageFromSPToIDPencryptKeyPKIpublicIDPagentAnonceAUTHAnonceAUTH
REPLYIDPnonceAUTHREADYSP (ProtocolClause msg)
 = case msg {Message( AgentVar( "SP" ),   AgentVar( "IDP" ),

       Cons (  Encrypt( KeyPKI(Public,AgentVar( "IDP" )),
       Cons (  Agent(AgentVar( "A" )),
       Cons ( Nonce( NonceVar ( "AUTH" ), AgentVar( "A" ) ),
       Cons ( Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ),
       Cons ( Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),  Nil
  )  )  )  )  ) ,  Nil
  )  )     => True ;

     _ => False ;

     }  ;



 class  IDPclass (AgentTerm name, Network network) implements Agent{
 Bool messageFromAToIDPagentAnonceANONCEAnonceSIDAtextCryptoOfferAQuery = False
 ;

   Bool messageFromAToIDPencryptKeyPKIpublicIDPnoncePMSAQuery = False  ;

   Bool messageFromAToIDPencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnonceP
MSAQuery = False  ;

   Bool messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPha
shnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMast
erSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPno
nceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptK
eyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSAQuery = False  ;

   Bool messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPha
shnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPnonce
DYNAMICPASSWORDAagentAnonceREGISTERAQuery = False  ;

   Bool messageFromAToIDPagentAnonce*ANONCEAnonce*SIDAtextCryptoOfferAQuery = Fa
lse  ;

   Bool messageFromAToIDPencryptKeyPKIpublicIDPnonce*PMSAQuery = False  ;

   Bool messageFromAToIDPencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce
*PMSAQuery = False  ;

   Bool messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDP
hashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSAte
xtMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*IDP
NONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonce*
PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSAQuery = False  ;


   Bool messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDP
hashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatetose
rviceagentAagentSPnonceAUTHAQuery = False  ;

   Bool messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDP
```

```
hashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatetose
rviceagentAnonceDYNAMICPASSWORDAnonceAUTHAnonceAUTHREPLYIDPQuery = False  ;

   Bool messageFromSPToIDPencryptKeyPKIpublicIDPagentAnonceAUTHAnonceAUTHREPLYID
PnonceAUTHREADYSPQuery = False  ;

    ProtocolClause messageFromAToIDPagentAnonceANONCEAnonceSIDAtextCryptoOfferA
=  UndefinedClause   ;

    PayloadElement agentA =  UndefinedElem   ;

    PayloadElement nonceANONCEA =  UndefinedElem   ;

    PayloadElement nonceSIDA =  UndefinedElem   ;

    PayloadElement textCryptoOfferA =  UndefinedElem   ;

    PayloadElement newnonceIDPNONCEIDP =  UndefinedElem   ;

    PayloadElement textCryptoOfferIDP =  UndefinedElem   ;

    ProtocolClause messageFromIDPToAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferIDP
=  UndefinedClause  ;

    ProtocolClause messageFromAToIDPencryptKeyPKIpublicIDPnoncePMSA =  Undefined
Clause   ;

    PayloadElement keyKeyPKIpublicIDP =  UndefinedElem   ;

    PayloadElement keyKeyPKIprivateIDP =  UndefinedElem   ;

    PayloadElement decryptKeyPKIprivateIDPencryptKeyPKIpublicIDPnoncePMSA =  Und
efinedElem   ;

    PayloadElement noncePMSA =  UndefinedElem   ;

    ProtocolClause messageFromAToIDPencryptKeyPKIprivateAhashnonceIDPNONCEIDPage
ntIDPnoncePMSA =  UndefinedClause   ;

    PayloadElement keyKeyPKIpublicA =  UndefinedElem   ;

    PayloadElement decryptKeyPKIpublicAencryptKeyPKIprivateAhashnonceIDPNONCEIDP
agentIDPnoncePMSA =  UndefinedElem   ;

    PayloadElement hashnonceIDPNONCEIDPagentIDPnoncePMSA =  UndefinedElem   ;


    ProtocolClause messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceID
PNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnonceP
MSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceID
PNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonceP
MSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =  UndefinedClause
   ;

    PayloadElement encryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnonc
ePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMasterSecr
etnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonceSID
AtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyPKIp
rivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =  UndefinedElem   ;

    PayloadElement textMasterSecret =  UndefinedElem   ;

    PayloadElement hashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDP =
UndefinedElem   ;

    PayloadElement agentIDP =  UndefinedElem   ;

    PayloadElement encryptKeyPKIpublicIDPnoncePMSA =  UndefinedElem   ;

    PayloadElement encryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =
 UndefinedElem   ;
```

```
    PayloadElement hashhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDP
agentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOff
erIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPage
ntIDPnoncePMSA =  UndefinedElem   ;

    PayloadElement keyKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashnonceP
MSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDP =  UndefinedElem   ;

    PayloadElement encryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashno
ncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMasterSe
cretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonceS
IDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyPK
IprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =  UndefinedElem   ;

    ProtocolClause messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonce
IDPNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnonc
ePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonce
IDPNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonc
ePMSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =  UndefinedClau
se  ;

    ProtocolClause messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceID
PNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegisterto
AS/IDPnonceDYNAMICPASSWORDAagentAnonceREGISTERA =  UndefinedClause   ;

    PayloadElement keyKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnoncePMS
AtextMasterSecretnonceANONCEAnonceIDPNONCEIDP =  UndefinedElem   ;

    PayloadElement decryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnonc
ePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPencryptKeyCompositeagentAnonceA
NONCEAnonceIDPNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPt
extRegistertoAS/IDPnonceDYNAMICPASSWORDAagentAnonceREGISTERA =  UndefinedElem
;

    PayloadElement textRegistertoAS/IDP =  UndefinedElem   ;

    PayloadElement nonceDYNAMICPASSWORDA =  UndefinedElem   ;

    PayloadElement nonceREGISTERA =  UndefinedElem   ;

    PayloadElement newnonceREPLYIDP =  UndefinedElem   ;

    PayloadElement encryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashno
ncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPagentAage
ntIDPnonceREGISTERAnonceREPLYIDP =  UndefinedElem   ;

    ProtocolClause messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonce
IDPNONCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegister
toAS/IDPagentAagentIDPnonceREGISTERAnonceREPLYIDP =  UndefinedClause  ;

    ProtocolClause messageFromAToIDPagentAnonce*ANONCEAnonce*SIDAtextCryptoOffer
A =  UndefinedClause   ;

    PayloadElement nonce*ANONCEA =  UndefinedElem   ;

    PayloadElement nonce*SIDA =  UndefinedElem   ;

    PayloadElement newnonce*IDPNONCEIDP =  UndefinedElem   ;

    ProtocolClause messageFromIDPToAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferID
P =  UndefinedClause  ;

    ProtocolClause messageFromAToIDPencryptKeyPKIpublicIDPnonce*PMSA =  Undefine
dClause   ;

    PayloadElement decryptKeyPKIprivateIDPencryptKeyPKIpublicIDPnonce*PMSA =  Un
definedElem   ;

    PayloadElement nonce*PMSA =  UndefinedElem   ;

    ProtocolClause messageFromAToIDPencryptKeyPKIprivateAhashnonce*IDPNONCEIDPag
entIDPnonce*PMSA =  UndefinedClause   ;
```

```
    PayloadElement decryptKeyPKIpublicAencryptKeyPKIprivateAhashnonce*IDPNONCEID
PagentIDPnonce*PMSA =  UndefinedElem   ;

    PayloadElement hashnonce*IDPNONCEIDPagentIDPnonce*PMSA =  UndefinedElem   ;


    ProtocolClause messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*
IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashn
once*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONC
EAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpubl
icIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA =  U
ndefinedClause   ;

    PayloadElement hashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDP
=  UndefinedElem   ;

    PayloadElement keyKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*
PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDP =  UndefinedElem   ;

    PayloadElement decryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashno
nce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPencryptKeyCompositeagentAn
once*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*ID
PNONCEIDPhashhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAa
gentIDPnonce*ANONCEAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferID
PencryptKeyPKIpublicIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagent
IDPnonce*PMSA =  UndefinedElem   ;

    PayloadElement hashhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCE
IDPagentAagentIDPnonce*ANONCEAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCry
ptoOfferIDPencryptKeyPKIpublicIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNON
CEIDPagentIDPnonce*PMSA =  UndefinedElem   ;

    PayloadElement keyKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonc
e*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDP =  UndefinedElem   ;

    PayloadElement encryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhash
nonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSAtextMa
sterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*IDPNONC
EIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonce*PMSA
encryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA =  UndefinedElem
;

    ProtocolClause messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonc
e*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhas
hnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANO
NCEAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpu
blicIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA =
 UndefinedClause  ;

    ProtocolClause messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*
IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthe
nticatetoserviceagentAagentSPnonceAUTHA =  UndefinedClause   ;

    PayloadElement decryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashno
nce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPencryptKeyCompositeagentAn
once*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*ID
PNONCEIDPtextauthenticatetoserviceagentAagentSPnonceAUTHA =  UndefinedElem   ;


    PayloadElement textauthenticatetoservice =  UndefinedElem   ;

    PayloadElement agentSP =  UndefinedElem   ;

    PayloadElement nonceAUTHA =  UndefinedElem   ;

    PayloadElement newnonceAUTHREPLYIDP =  UndefinedElem   ;

    PayloadElement encryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhash
nonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextRegistertoAS/IDPagen
tAagentIDPnonceAUTHAnonceAUTHREPLYIDP =  UndefinedElem   ;
```

```
    ProtocolClause messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonc
e*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextReg
istertoAS/IDPagentAagentIDPnonceAUTHAnonceAUTHREPLYIDP =  UndefinedClause  ;

    ProtocolClause messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*
IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthe
nticatetoserviceagentAnonceDYNAMICPASSWORDAnonceAUTHAnonceAUTHREPLYIDP =  Undefi
nedClause   ;

    PayloadElement decryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashno
nce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPencryptKeyCompositeagentAn
once*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*ID
PNONCEIDPtextauthenticatetoserviceagentAnonceDYNAMICPASSWORDAnonceAUTHAnonceAUTH
REPLYIDP =  UndefinedElem   ;

    PayloadElement hashnonceDYNAMICPASSWORDA =  UndefinedElem   ;

    PayloadElement newkeyKeyPKIprivateSP =  UndefinedElem   ;

    PayloadElement newkeyKeyPKIpublicSP =  UndefinedElem   ;

    PayloadElement encryptKeyPKIpublicSPagentAnonceAUTHAnonceAUTHREPLYIDPhashnon
ceDYNAMICPASSWORDA =  UndefinedElem   ;

    ProtocolClause messageFromIDPToSPencryptKeyPKIpublicSPagentAnonceAUTHAnonceA
UTHREPLYIDPhashnonceDYNAMICPASSWORDA =  UndefinedClause  ;

    ProtocolClause messageFromSPToIDPencryptKeyPKIpublicIDPagentAnonceAUTHAnonce
AUTHREPLYIDPnonceAUTHREADYSP =  UndefinedClause   ;

    PayloadElement decryptKeyPKIprivateIDPencryptKeyPKIpublicIDPagentAnonceAUTHA
nonceAUTHREPLYIDPnonceAUTHREADYSP =  UndefinedElem   ;

    PayloadElement nonceAUTHREADYSP =  UndefinedElem   ;

    PayloadElement encryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhash
nonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPnonceAUTHAnonceAUTHREPLY
IDPnonceAUTHREADYSP =  UndefinedElem   ;

    ProtocolClause messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonc
e*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPnonceAU
THAnonceAUTHREPLYIDPnonceAUTHREADYSP =  UndefinedClause  ;

      Unit run(){ await(network != null) ;

     await( messageFromAToIDPagentAnonceANONCEAnonceSIDAtextCryptoOfferAQuery )
 ;

   messageFromAToIDPagentAnonceANONCEAnonceSIDAtextCryptoOfferA = Message(AgentV
ar( "A" ),AgentVar( "IDP" ),
  list[  Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) )
, Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("CryptoOffer", Agent
Var( "A" )) ]) ;

   agentA =  Agent(AgentVar( "A" )) ;

   nonceANONCEA = Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ) ;

   nonceSIDA = Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ) ;

   textCryptoOfferA =  TextOrigin("CryptoOffer", AgentVar( "A" ))  ;

   newnonceIDPNONCEIDP =  New(Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" )
) ) ;

   textCryptoOfferIDP =  TextOrigin("CryptoOffer", AgentVar( "IDP" ))  ;

   messageFromIDPToAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferIDP = Message(AgentV
ar( "IDP" ),AgentVar( "A" ),
  list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "
SID" ), AgentVar( "A" ) ),  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ]) ;
```

```
   network!send( Message(AgentVar( "IDP" ),AgentVar( "A" ),
  list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "
SID" ), AgentVar( "A" ) ),  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ]) )  ;


   await( messageFromAToIDPencryptKeyPKIpublicIDPnoncePMSAQuery )  ;

   messageFromAToIDPencryptKeyPKIpublicIDPnoncePMSA = Message(AgentVar( "A" ),Ag
entVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] )]) ;

   keyKeyPKIpublicIDP =  Key( KeyPKI(Public,AgentVar( "IDP" ))) ;

   keyKeyPKIprivateIDP =  Key( KeyPKI(Private,AgentVar( "IDP" ))) ;

   decryptKeyPKIprivateIDPencryptKeyPKIpublicIDPnoncePMSA =
  Decrypt(   KeyPKI(Private,AgentVar( "IDP" )), list[
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ) ] ) ;

   noncePMSA = Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ) ;

   await( messageFromAToIDPencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnonc
ePMSAQuery )  ;

   messageFromAToIDPencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =
 Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )]) ;

   keyKeyPKIpublicA =  Key( KeyPKI(Public,AgentVar( "A" ))) ;

   decryptKeyPKIpublicAencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMS
A =
  Decrypt(   KeyPKI(Public,AgentVar( "A" )), list[
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] ) ] ) ;

   hashnonceIDPNONCEIDPagentIDPnoncePMSA =
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] ) ;

   await( messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDP
hashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMa
sterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDP
nonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryp
tKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSAQuery )  ;

   messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnon
cePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMasterSec
retnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonceSI
DAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyPKI
privateAhashnonceIDPNONCEIDPagentIDPnoncePMSA = Message(AgentVar( "A" ),AgentVar
( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONC
E" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
```

```
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )] )] )]) ;

   encryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMasterS
ecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMasterSecretnonceANONCEAno
nceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonceSIDAtextCryptoOffer
AtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyPKIprivateAhashnonce
IDPNONCEIDPagentIDPnoncePMSA =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONC
E" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )] ) ;

   textMasterSecret = Text("MasterSecret") ;

   hashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDP =
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ) ;

   agentIDP =  Agent(AgentVar( "IDP" )) ;

   encryptKeyPKIpublicIDPnoncePMSA =
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ) ;

   encryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] ) ;

   hashhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPno
nceANONCEAnonceIDPNONCEIDPnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyP
KIpublicIDPnoncePMSAencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA =

 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] ) ;

   keyKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMasterSec
retnonceANONCEAnonceIDPNONCEIDP =  Key( KeyComposite(list[   Agent(AgentVar( "ID
P" )), Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNO
NCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) ) ;
```

```
   encryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMaste
rSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMasterSecretnonceANONCEA
nonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonceSIDAtextCryptoOff
erAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyPKIprivateAhashnon
ceIDPNONCEIDPagentIDPnoncePMSA =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "ANO
NCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )] ) ;

   messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashn
oncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMasterS
ecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonce
SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyP
KIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA = Message(AgentVar( "IDP" ),Agen
tVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "ANO
NCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )] )]) ;

   network!send( Message(AgentVar( "IDP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "ANO
NCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )), No
nce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), A
gentVar( "IDP" ) ), Nonce( NonceVar ( "SID" ), AgentVar( "A" ) ),  TextOrigin("C
ryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "PMS" ), Ag
entVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentVa
r( "IDP" )), Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) )] )] )] )]) )  ;

   await( messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDP
hashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPnon
ceDYNAMICPASSWORDAagentAnonceREGISTERAQuery )  ;
```

```
   messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnon
cePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPnonceDYNAM
ICPASSWORDAagentAnonceREGISTERA = Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONC
E" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"), Nonce( NonceVar
( "DYNAMICPASSWORD" ), AgentVar( "A" ) ),  Agent(AgentVar( "A" )), Nonce( NonceV
ar ( "REGISTER" ), AgentVar( "A" ) )] )]) ;

   keyKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMasterSecre
tnonceANONCEAnonceIDPNONCEIDP =  Key( KeyComposite(list[   Agent(AgentVar( "A" )
), Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE"
 ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) ) ;

   decryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMasterS
ecretnonceANONCEAnonceIDPNONCEIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNO
NCEIDPhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/
IDPnonceDYNAMICPASSWORDAagentAnonceREGISTERA =
  Decrypt(   KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANO
NCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "ANONC
E" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"), Nonce( NonceVar
( "DYNAMICPASSWORD" ), AgentVar( "A" ) ),  Agent(AgentVar( "A" )), Nonce( NonceV
ar ( "REGISTER" ), AgentVar( "A" ) )] ) ] ) ;

   textRegistertoAS/IDP = Text("Register to AS/IDP") ;

   nonceDYNAMICPASSWORDA = Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A"
) ) ;

   nonceREGISTERA = Nonce( NonceVar ( "REGISTER" ), AgentVar( "A" ) ) ;

   newnonceREPLYIDP =  New(Nonce( NonceVar ( "REPLY" ), AgentVar( "IDP" ) ) ) ;


   encryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashnoncePMSAtextMaste
rSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPagentAagentIDPnonceREGIST
ERAnonceREPLYIDP =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "ANO
NCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"),  Agent(AgentVar(
 "A" )),  Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "REGISTER" ), AgentVar( "A
" ) ), Nonce( NonceVar ( "REPLY" ), AgentVar( "IDP" ) )] ) ;

   messageFromIDPToAencryptKeyCompositeagentIDPnonceANONCEAnonceIDPNONCEIDPhashn
oncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPagentAag
entIDPnonceREGISTERAnonceREPLYIDP = Message(AgentVar( "IDP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "ANO
NCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"),  Agent(AgentVar(
 "A" )),  Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "REGISTER" ), AgentVar( "A
" ) ), Nonce( NonceVar ( "REPLY" ), AgentVar( "IDP" ) )] )]) ;

   network!send( Message(AgentVar( "IDP" ),AgentVar( "A" ),
```

```
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "ANO
NCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "PMS" ), AgentVar( "A" ) ), Text("MasterSecret"),
 Nonce( NonceVar ( "ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "IDPNONCE" )
, AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"),  Agent(AgentVar(
 "A" )),  Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "REGISTER" ), AgentVar( "A
" ) ), Nonce( NonceVar ( "REPLY" ), AgentVar( "IDP" ) )] )]) )  ;

   await( messageFromAToIDPagentAnonce*ANONCEAnonce*SIDAtextCryptoOfferAQuery )
 ;

   messageFromAToIDPagentAnonce*ANONCEAnonce*SIDAtextCryptoOfferA = Message(Agen
tVar( "A" ),AgentVar( "IDP" ),
  list[  Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" )
), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOrigin("CryptoOffer", Age
ntVar( "A" )) ]) ;

   nonce*ANONCEA = Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ) ;

   nonce*SIDA = Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ) ;

   newnonce*IDPNONCEIDP =  New(Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP"
) ) ) ;

   messageFromIDPToAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferIDP = Message(Agen
tVar( "IDP" ),AgentVar( "A" ),
  list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ), Nonce( NonceVar (
"*SID" ), AgentVar( "A" ) ),  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ]) ;


   network!send( Message(AgentVar( "IDP" ),AgentVar( "A" ),
  list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ), Nonce( NonceVar (
"*SID" ), AgentVar( "A" ) ),  TextOrigin("CryptoOffer", AgentVar( "IDP" )) ]) )
 ;

   await( messageFromAToIDPencryptKeyPKIpublicIDPnonce*PMSAQuery )  ;

   messageFromAToIDPencryptKeyPKIpublicIDPnonce*PMSA = Message(AgentVar( "A" ),A
gentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] )]) ;

   decryptKeyPKIprivateIDPencryptKeyPKIpublicIDPnonce*PMSA =
  Decrypt(   KeyPKI(Private,AgentVar( "IDP" )), list[
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ) ] ) ;

   nonce*PMSA = Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ) ;

   await( messageFromAToIDPencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnon
ce*PMSAQuery )  ;

   messageFromAToIDPencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA
 = Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )]) ;

   decryptKeyPKIpublicAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*P
MSA =
  Decrypt(   KeyPKI(Public,AgentVar( "A" )), list[
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] ) ] ) ;

   hashnonce*IDPNONCEIDPagentIDPnonce*PMSA =
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] ) ;
```

```
   await( messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEI
DPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSA
textMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*I
DPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonc
e*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSAQuery )  ;

   messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashn
once*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSAtextMas
terSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*IDPNONCE
IDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonce*PMSAe
ncryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA = Message(AgentVar(
"A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )] )]) ;

   hashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDP =
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ) ;

   keyKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMasterSe
cretnonce*ANONCEAnonce*IDPNONCEIDP =  Key( KeyComposite(list[   Agent(AgentVar(
"A" )), Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*ID
PNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) ) ;

   decryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMast
erSecretnonce*ANONCEAnonce*IDPNONCEIDPencryptKeyCompositeagentAnonce*ANONCEAnonc
e*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhas
hnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANO
NCEAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpu
blicIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA =
  Decrypt(   KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
```

```
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )] ) ] ) ;


   hashhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentID
Pnonce*ANONCEAnonce*IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencry
ptKeyPKIpublicIDPnonce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnon
ce*PMSA =
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] ) ;

   keyKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMaster
Secretnonce*ANONCEAnonce*IDPNONCEIDP =  Key( KeyComposite(list[   Agent(AgentVar
( "IDP" )), Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar (
"*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) ) ;

   encryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMa
sterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSAtextMasterSecretnonce*
ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*IDPNONCEIDPnonce*SIDAte
xtCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonce*PMSAencryptKeyPKIpri
vateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )] ) ;

   messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhas
hnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSAtextM
asterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*IDPNON
CEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonce*PMS
AencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA = Message(AgentVar
( "IDP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
```

```
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )] )] )]) ;

   network!send( Message(AgentVar( "IDP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
 Hash( list[
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] ),  Agent(AgentVar( "A" )),  Agent(AgentVar( "IDP" )),
 Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE"
 ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "*SID" ), AgentVar( "A" ) ),  TextOri
gin("CryptoOffer", AgentVar( "A" )) ,  TextOrigin("CryptoOffer", AgentVar( "IDP"
 )) ,
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[ Nonce( NonceVar ( "*PMS" ), A
gentVar( "A" ) )] ),
  Encrypt( KeyPKI(Private,AgentVar( "A" )), list[
 Hash( list[ Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),  Agent(AgentV
ar( "IDP" )), Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) )] )] )] )] )]) )  ;


   await( messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEI
DPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticateto
serviceagentAagentSPnonceAUTHAQuery )  ;

   messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashn
once*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatetoservice
agentAagentSPnonceAUTHA = Message(AgentVar( "A" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("authenticate to service"),  Agent(A
gentVar( "A" )),  Agent(AgentVar( "SP" )), Nonce( NonceVar ( "AUTH" ), AgentVar(
 "A" ) )] )]) ;

   decryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMast
erSecretnonce*ANONCEAnonce*IDPNONCEIDPencryptKeyCompositeagentAnonce*ANONCEAnonc
e*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextaut
henticatetoserviceagentAagentSPnonceAUTHA =
  Decrypt(   KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("authenticate to service"),  Agent(A
gentVar( "A" )),  Agent(AgentVar( "SP" )), Nonce( NonceVar ( "AUTH" ), AgentVar(
 "A" ) )] ) ] ) ;

   textauthenticatetoservice = Text("authenticate to service") ;

   agentSP =  Agent(AgentVar( "SP" )) ;
```

```
   nonceAUTHA = Nonce( NonceVar ( "AUTH" ), AgentVar( "A" ) ) ;

   newnonceAUTHREPLYIDP =  New(Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP"
) ) ) ;

   encryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMa
sterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextRegistertoAS/IDPagentAagentIDPnonceA
UTHAnonceAUTHREPLYIDP =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"),  Agent(AgentV
ar( "A" )),  Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "AUTH" ), AgentVar( "A"
 ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) )] ) ;

   messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhas
hnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextRegistertoAS/IDPage
ntAagentIDPnonceAUTHAnonceAUTHREPLYIDP = Message(AgentVar( "IDP" ),AgentVar( "A"
 ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"),  Agent(AgentV
ar( "A" )),  Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "AUTH" ), AgentVar( "A"
 ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) )] )]) ;

   network!send( Message(AgentVar( "IDP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("Register to AS/IDP"),  Agent(AgentV
ar( "A" )),  Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "AUTH" ), AgentVar( "A"
 ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) )] )]) )  ;

   await( messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEI
DPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticateto
serviceagentAnonceDYNAMICPASSWORDAnonceAUTHAnonceAUTHREPLYIDPQuery )  ;

   messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashn
once*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatetoservice
agentAnonceDYNAMICPASSWORDAnonceAUTHAnonceAUTHREPLYIDP = Message(AgentVar( "A" )
,AgentVar( "IDP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("authenticate to service"),  Agent(A
gentVar( "A" )), Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) ), Nonce
( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agent
Var( "IDP" ) )] )]) ;

   decryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMast
erSecretnonce*ANONCEAnonce*IDPNONCEIDPencryptKeyCompositeagentAnonce*ANONCEAnonc
e*IDPNONCEIDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextaut
henticatetoserviceagentAnonceDYNAMICPASSWORDAnonceAUTHAnonceAUTHREPLYIDP =
  Decrypt(   KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "*ANON
CE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) ),
```

```
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Text("authenticate to service"),  Agent(A
gentVar( "A" )), Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) ), Nonce
( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agent
Var( "IDP" ) )] ) ] ) ;

   hashnonceDYNAMICPASSWORDA =
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] ) ;

   newkeyKeyPKIprivateSP =  New(Key( KeyPKI(Private,AgentVar( "SP" )))) ;

   newkeyKeyPKIpublicSP =  New(Key( KeyPKI(Public,AgentVar( "SP" )))) ;

   encryptKeyPKIpublicSPagentAnonceAUTHAnonceAUTHREPLYIDPhashnonceDYNAMICPASSWOR
DA =
  Encrypt( KeyPKI(Public,AgentVar( "SP" )), list[  Agent(AgentVar( "A" )), Nonce
( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agent
Var( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) ;


   messageFromIDPToSPencryptKeyPKIpublicSPagentAnonceAUTHAnonceAUTHREPLYIDPhashn
onceDYNAMICPASSWORDA = Message(AgentVar( "IDP" ),AgentVar( "SP" ),
  list[
  Encrypt( KeyPKI(Public,AgentVar( "SP" )), list[  Agent(AgentVar( "A" )), Nonce
( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agent
Var( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] )]) ;


   network!send( Message(AgentVar( "IDP" ),AgentVar( "SP" ),
  list[
  Encrypt( KeyPKI(Public,AgentVar( "SP" )), list[  Agent(AgentVar( "A" )), Nonce
( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agent
Var( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] )]) )
 ;

   await( messageFromSPToIDPencryptKeyPKIpublicIDPagentAnonceAUTHAnonceAUTHREPLY
IDPnonceAUTHREADYSPQuery )  ;

   messageFromSPToIDPencryptKeyPKIpublicIDPagentAnonceAUTHAnonceAUTHREPLYIDPnonc
eAUTHREADYSP = Message(AgentVar( "SP" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[  Agent(AgentVar( "A" )), Nonc
e( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agen
tVar( "IDP" ) ), Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) )] )]) ;

   decryptKeyPKIprivateIDPencryptKeyPKIpublicIDPagentAnonceAUTHAnonceAUTHREPLYID
PnonceAUTHREADYSP =
  Decrypt(   KeyPKI(Private,AgentVar( "IDP" )), list[
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[  Agent(AgentVar( "A" )), Nonc
e( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agen
tVar( "IDP" ) ), Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) )] ) ] ) ;


   nonceAUTHREADYSP = Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ) ;

   encryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhashnonce*PMSAtextMa
sterSecretnonce*ANONCEAnonce*IDPNONCEIDPnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREAD
YSP =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Nonce( NonceVar ( "AUTH" ), AgentVar( "A"
 ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "A
UTHREADY" ), AgentVar( "SP" ) )] ) ;

   messageFromIDPToAencryptKeyCompositeagentIDPnonce*ANONCEAnonce*IDPNONCEIDPhas
```

```
hnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPnonceAUTHAnonceAUTHREPL
YIDPnonceAUTHREADYSP = Message(AgentVar( "IDP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Nonce( NonceVar ( "AUTH" ), AgentVar( "A"
 ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "A
UTHREADY" ), AgentVar( "SP" ) )] )]) ;

   network!send( Message(AgentVar( "IDP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "IDP" )), Nonce( NonceVar ( "*AN
ONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE" ), AgentVar( "IDP" ) )
,
 Hash( list[ Nonce( NonceVar ( "*PMS" ), AgentVar( "A" ) ), Text("MasterSecret")
, Nonce( NonceVar ( "*ANONCE" ), AgentVar( "A" ) ), Nonce( NonceVar ( "*IDPNONCE
" ), AgentVar( "IDP" ) )] )] ) , list[ Nonce( NonceVar ( "AUTH" ), AgentVar( "A"
 ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), Nonce( NonceVar ( "A
UTHREADY" ), AgentVar( "SP" ) )] )]) )  ;

     }
   /* End of run-method */


    Unit receive(ProtocolClause msg){
  if ( testmessageFromAToIDPagentAnonceANONCEAnonceSIDAtextCryptoOfferA (msg) )
  {messageFromAToIDPagentAnonceANONCEAnonceSIDAtextCryptoOfferAQuery = True ;


     }
     else  {
  if ( testmessageFromAToIDPencryptKeyPKIpublicIDPnoncePMSA (msg) )
  {messageFromAToIDPencryptKeyPKIpublicIDPnoncePMSAQuery = True ;

     }
     else  {
  if ( testmessageFromAToIDPencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnon
cePMSA (msg) )
  {messageFromAToIDPencryptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSAQu
ery = True ;

     }
     else  {
  if ( testmessageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEID
PhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextM
asterSecretnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEID
PnonceSIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencry
ptKeyPKIprivateAhashnonceIDPNONCEIDPagentIDPnoncePMSA (msg) )
  {messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnon
cePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPhashhashnoncePMSAtextMasterSec
retnonceANONCEAnonceIDPNONCEIDPagentAagentIDPnonceANONCEAnonceIDPNONCEIDPnonceSI
DAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnoncePMSAencryptKeyPKI
privateAhashnonceIDPNONCEIDPagentIDPnoncePMSAQuery = True ;

     }
     else  {
  if ( testmessageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEID
PhashnoncePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPno
nceDYNAMICPASSWORDAagentAnonceREGISTERA (msg) )
  {messageFromAToIDPencryptKeyCompositeagentAnonceANONCEAnonceIDPNONCEIDPhashnon
cePMSAtextMasterSecretnonceANONCEAnonceIDPNONCEIDPtextRegistertoAS/IDPnonceDYNAM
ICPASSWORDAagentAnonceREGISTERAQuery = True ;

     }
     else  {
  if ( testmessageFromAToIDPagentAnonce*ANONCEAnonce*SIDAtextCryptoOfferA (msg)
)
  {messageFromAToIDPagentAnonce*ANONCEAnonce*SIDAtextCryptoOfferAQuery = True ;
```

```
     }
     else  {
  if ( testmessageFromAToIDPencryptKeyPKIpublicIDPnonce*PMSA (msg) )
  {messageFromAToIDPencryptKeyPKIpublicIDPnonce*PMSAQuery = True ;

     }
     else  {
  if ( testmessageFromAToIDPencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPno
nce*PMSA (msg) )
  {messageFromAToIDPencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA
Query = True ;

     }
     else  {
  if ( testmessageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCE
IDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMS
AtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*
IDPNONCEIDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnon
ce*PMSAencryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSA (msg) )
  {messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashn
once*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPhashhashnonce*PMSAtextMas
terSecretnonce*ANONCEAnonce*IDPNONCEIDPagentAagentIDPnonce*ANONCEAnonce*IDPNONCE
IDPnonce*SIDAtextCryptoOfferAtextCryptoOfferIDPencryptKeyPKIpublicIDPnonce*PMSAe
ncryptKeyPKIprivateAhashnonce*IDPNONCEIDPagentIDPnonce*PMSAQuery = True ;

     }
     else  {
  if ( testmessageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCE
IDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatet
oserviceagentAagentSPnonceAUTHA (msg) )
  {messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashn
once*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatetoservice
agentAagentSPnonceAUTHAQuery = True ;

     }
     else  {
  if ( testmessageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCE
IDPhashnonce*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatet
oserviceagentAnonceDYNAMICPASSWORDAnonceAUTHAnonceAUTHREPLYIDP (msg) )
  {messageFromAToIDPencryptKeyCompositeagentAnonce*ANONCEAnonce*IDPNONCEIDPhashn
once*PMSAtextMasterSecretnonce*ANONCEAnonce*IDPNONCEIDPtextauthenticatetoservice
agentAnonceDYNAMICPASSWORDAnonceAUTHAnonceAUTHREPLYIDPQuery = True ;

     }
     else  {
  if ( testmessageFromSPToIDPencryptKeyPKIpublicIDPagentAnonceAUTHAnonceAUTHREPL
YIDPnonceAUTHREADYSP (msg) )
  {messageFromSPToIDPencryptKeyPKIpublicIDPagentAnonceAUTHAnonceAUTHREPLYIDPnonc
eAUTHREADYSPQuery = True ;

     }
     else  { skip  ;

       }
   /* End of test method */

     }
   /* End of test method */

     }
   /* End of test method */

     }
   /* End of test method */

     }
   /* End of test method */

     }
   /* End of test method */
```

```
    }
  /* End of test method */

    }
  /* End of test method */

    }
  /* End of test method */

    }
  /* End of test method */

    }
  /* End of test method */

    }
  /* End of test method */

 }
  /* End of receive method */

 }
 /* End of class */

 def Bool testmessageFromIDPToSPencryptKeyPKIpublicSPagentAnonceAUTHAnonceAUTHR
EPLYIDPhashnonceDYNAMICPASSWORDA (ProtocolClause msg)
 = case msg {Message( AgentVar( "IDP" ),   AgentVar( "SP" ),

      Cons (  Encrypt( KeyPKI(Public,AgentVar( "SP" )),
      Cons (  Agent(AgentVar( "A" )),
      Cons ( Nonce( NonceVar ( "AUTH" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) ),  Nil
 )  ) ,  Nil
 )  )  )  )  ) ,  Nil
 )  )     => True ;

    _ => False ;

    }  ;


 def Bool testmessageFromAToSPencryptKeyCompositeagentAnonceAUTHAnonceAUTHREPLY
IDPnonceAUTHREADYSPhashnonceDYNAMICPASSWORDAtextRequestServicenonceSESSIONIDA (P
rotocolClause msg)
 = case msg {Message( AgentVar( "A" ),   AgentVar( "SP" ),

      Cons (  Encrypt(KeyComposite (
      Cons (  Agent(AgentVar( "A" )),
      Cons ( Nonce( NonceVar ( "AUTH" ), AgentVar( "A" ) ),
      Cons ( Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ),
      Cons ( Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
      Cons (  Hash(
      Cons ( Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) ),  Nil
 )  ) ,  Nil
 )  )  )  )  ) ),
      Cons ( Text("Request Service"),
      Cons ( Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) ),  Nil
 )  )  ) ,  Nil
 )  )     => True ;

    _ => False ;

    }  ;



 class  SPclass (AgentTerm name, Network network) implements Agent{
 Bool messageFromIDPToSPencryptKeyPKIpublicSPagentAnonceAUTHAnonceAUTHREPLYIDPha
shnonceDYNAMICPASSWORDAQuery = False  ;
```

```
   Bool messageFromAToSPencryptKeyCompositeagentAnonceAUTHAnonceAUTHREPLYIDPnonc
eAUTHREADYSPhashnonceDYNAMICPASSWORDAtextRequestServicenonceSESSIONIDAQuery = Fa
lse  ;

    ProtocolClause messageFromIDPToSPencryptKeyPKIpublicSPagentAnonceAUTHAnonceA
UTHREPLYIDPhashnonceDYNAMICPASSWORDA =  UndefinedClause   ;

    PayloadElement agentIDP =  UndefinedElem   ;

    PayloadElement keyKeyPKIpublicSP =  UndefinedElem   ;

    PayloadElement keyKeyPKIprivateSP =  UndefinedElem   ;

    PayloadElement decryptKeyPKIprivateSPencryptKeyPKIpublicSPagentAnonceAUTHAno
nceAUTHREPLYIDPhashnonceDYNAMICPASSWORDA =  UndefinedElem   ;

    PayloadElement agentA =  UndefinedElem   ;

    PayloadElement nonceAUTHA =  UndefinedElem   ;

    PayloadElement nonceAUTHREPLYIDP =  UndefinedElem   ;

    PayloadElement hashnonceDYNAMICPASSWORDA =  UndefinedElem   ;

    PayloadElement newnonceAUTHREADYSP =  UndefinedElem   ;

    PayloadElement keyKeyPKIpublicIDP =  UndefinedElem   ;

    PayloadElement encryptKeyPKIpublicIDPagentAnonceAUTHAnonceAUTHREPLYIDPnonceA
UTHREADYSP =  UndefinedElem   ;

    ProtocolClause messageFromSPToIDPencryptKeyPKIpublicIDPagentAnonceAUTHAnonce
AUTHREPLYIDPnonceAUTHREADYSP =  UndefinedClause  ;

    ProtocolClause messageFromAToSPencryptKeyCompositeagentAnonceAUTHAnonceAUTHR
EPLYIDPnonceAUTHREADYSPhashnonceDYNAMICPASSWORDAtextRequestServicenonceSESSIONID
A =  UndefinedClause   ;

    PayloadElement encryptKeyCompositeagentAnonceAUTHAnonceAUTHREPLYIDPnonceAUTH
READYSPhashnonceDYNAMICPASSWORDAtextRequestServicenonceSESSIONIDA =  UndefinedEl
em   ;

    PayloadElement textDeliverService =  UndefinedElem   ;

    PayloadElement nonceSESSIONIDA =  UndefinedElem   ;

    PayloadElement agentSP =  UndefinedElem   ;

    PayloadElement keyKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREA
DYSPhashnonceDYNAMICPASSWORDA =  UndefinedElem   ;

    PayloadElement encryptKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnonceAUT
HREADYSPhashnonceDYNAMICPASSWORDAtextDeliverServicenonceSESSIONIDA =  UndefinedE
lem   ;

    ProtocolClause messageFromSPToAencryptKeyCompositeagentSPnonceAUTHAnonceAUTH
REPLYIDPnonceAUTHREADYSPhashnonceDYNAMICPASSWORDAtextDeliverServicenonceSESSIONI
DA =  UndefinedClause  ;

      Unit run(){ await(network != null) ;

     await( messageFromIDPToSPencryptKeyPKIpublicSPagentAnonceAUTHAnonceAUTHREPL
YIDPhashnonceDYNAMICPASSWORDAQuery )  ;

   messageFromIDPToSPencryptKeyPKIpublicSPagentAnonceAUTHAnonceAUTHREPLYIDPhashn
onceDYNAMICPASSWORDA = Message(AgentVar( "IDP" ),AgentVar( "SP" ),
  list[
  Encrypt( KeyPKI(Public,AgentVar( "SP" )), list[  Agent(AgentVar( "A" )), Nonce
( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agent
Var( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] )]) ;
```

```
   agentIDP =  Agent(AgentVar( "IDP" )) ;

   keyKeyPKIpublicSP =  Key( KeyPKI(Public,AgentVar( "SP" ))) ;

   keyKeyPKIprivateSP =  Key( KeyPKI(Private,AgentVar( "SP" ))) ;

   decryptKeyPKIprivateSPencryptKeyPKIpublicSPagentAnonceAUTHAnonceAUTHREPLYIDPh
ashnonceDYNAMICPASSWORDA =
  Decrypt(   KeyPKI(Private,AgentVar( "SP" )), list[
  Encrypt( KeyPKI(Public,AgentVar( "SP" )), list[  Agent(AgentVar( "A" )), Nonce
( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agent
Var( "IDP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) ] )
;

   agentA =  Agent(AgentVar( "A" )) ;

   nonceAUTHA = Nonce( NonceVar ( "AUTH" ), AgentVar( "A" ) ) ;

   nonceAUTHREPLYIDP = Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ) ;


   hashnonceDYNAMICPASSWORDA =
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] ) ;

   newnonceAUTHREADYSP =  New(Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" )
) ) ;

   keyKeyPKIpublicIDP =  Key( KeyPKI(Public,AgentVar( "IDP" ))) ;

   encryptKeyPKIpublicIDPagentAnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREADYSP =
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[  Agent(AgentVar( "A" )), Nonc
e( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agen
tVar( "IDP" ) ), Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) )] ) ;

   messageFromSPToIDPencryptKeyPKIpublicIDPagentAnonceAUTHAnonceAUTHREPLYIDPnonc
eAUTHREADYSP = Message(AgentVar( "SP" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[  Agent(AgentVar( "A" )), Nonc
e( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agen
tVar( "IDP" ) ), Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) )] )]) ;

   network!send( Message(AgentVar( "SP" ),AgentVar( "IDP" ),
  list[
  Encrypt( KeyPKI(Public,AgentVar( "IDP" )), list[  Agent(AgentVar( "A" )), Nonc
e( NonceVar ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), Agen
tVar( "IDP" ) ), Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) )] )]) )  ;


   await( messageFromAToSPencryptKeyCompositeagentAnonceAUTHAnonceAUTHREPLYIDPno
nceAUTHREADYSPhashnonceDYNAMICPASSWORDAtextRequestServicenonceSESSIONIDAQuery )
 ;

   messageFromAToSPencryptKeyCompositeagentAnonceAUTHAnonceAUTHREPLYIDPnonceAUTH
READYSPhashnonceDYNAMICPASSWORDAtextRequestServicenonceSESSIONIDA = Message(Agen
tVar( "A" ),AgentVar( "SP" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "AUTH"
 ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), Non
ce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) , li
st[ Text("Request Service"), Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) )]
 )]) ;

   encryptKeyCompositeagentAnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREADYSPhashnonce
DYNAMICPASSWORDAtextRequestServicenonceSESSIONIDA =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "A" )), Nonce( NonceVar ( "AUTH"
 ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), Non
ce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) , li
```

```
st[ Text("Request Service"), Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) )]
 ) ;

   textDeliverService = Text("Deliver Service") ;

   nonceSESSIONIDA = Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) ) ;

   agentSP =  Agent(AgentVar( "SP" )) ;

   keyKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREADYSPhashnonceDYN
AMICPASSWORDA =  Key( KeyComposite(list[   Agent(AgentVar( "SP" )), Nonce( Nonce
Var ( "AUTH" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "I
DP" ) ), Nonce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) ) ;


   encryptKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnonceAUTHREADYSPhashnonc
eDYNAMICPASSWORDAtextDeliverServicenonceSESSIONIDA =
  Encrypt( KeyComposite(list[   Agent(AgentVar( "SP" )), Nonce( NonceVar ( "AUTH
" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), No
nce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) , li
st[ Text("Deliver Service"), Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) )]
 ) ;

   messageFromSPToAencryptKeyCompositeagentSPnonceAUTHAnonceAUTHREPLYIDPnonceAUT
HREADYSPhashnonceDYNAMICPASSWORDAtextDeliverServicenonceSESSIONIDA = Message(Age
ntVar( "SP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "SP" )), Nonce( NonceVar ( "AUTH
" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), No
nce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) , li
st[ Text("Deliver Service"), Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) )]
 )]) ;

   network!send( Message(AgentVar( "SP" ),AgentVar( "A" ),
  list[
  Encrypt( KeyComposite(list[   Agent(AgentVar( "SP" )), Nonce( NonceVar ( "AUTH
" ), AgentVar( "A" ) ), Nonce( NonceVar ( "AUTHREPLY" ), AgentVar( "IDP" ) ), No
nce( NonceVar ( "AUTHREADY" ), AgentVar( "SP" ) ),
 Hash( list[ Nonce( NonceVar ( "DYNAMICPASSWORD" ), AgentVar( "A" ) )] )] ) , li
st[ Text("Deliver Service"), Nonce( NonceVar ( "SESSIONID" ), AgentVar( "A" ) )]
 )]) )  ;

     }
   /* End of run-method */


    Unit receive(ProtocolClause msg){
  if ( testmessageFromIDPToSPencryptKeyPKIpublicSPagentAnonceAUTHAnonceAUTHREPLY
IDPhashnonceDYNAMICPASSWORDA (msg) )
  {messageFromIDPToSPencryptKeyPKIpublicSPagentAnonceAUTHAnonceAUTHREPLYIDPhashn
onceDYNAMICPASSWORDAQuery = True ;

     }
     else  {
  if ( testmessageFromAToSPencryptKeyCompositeagentAnonceAUTHAnonceAUTHREPLYIDPn
onceAUTHREADYSPhashnonceDYNAMICPASSWORDAtextRequestServicenonceSESSIONIDA (msg)
)
  {messageFromAToSPencryptKeyCompositeagentAnonceAUTHAnonceAUTHREPLYIDPnonceAUTH
READYSPhashnonceDYNAMICPASSWORDAtextRequestServicenonceSESSIONIDAQuery = True ;


     }
     else  { skip  ;

       }
   /* End of test method */

     }
   /* End of test method */
```

```
 }
  /* End of receive method */

 }
 /* End of class */



// The protocol: TLS version 1.1 withoutCertificatesexchangeCredTLS version 1.1
withoutCertificatesAuthenticationCredTransaction
{
Agent zombieAgent =  new cog ZombieAgent(AgentName("TestZombie") )  ;

  Network network  ;

  network =  new cog Network(zombieAgent)  ;

  Agent objectA ;

  objectA = new cog Aclass (AgentVar( "A" ) , network) ;

  network!register(AgentVar( "A" ), objectA)  ;

  Agent objectIDP ;

  objectIDP = new cog IDPclass (AgentVar( "IDP" ) , network) ;

  network!register(AgentVar( "IDP" ), objectIDP)  ;

  Agent objectSP ;

  objectSP = new cog SPclass (AgentVar( "SP" ) , network) ;

  network!register(AgentVar( "SP" ), objectSP)  ;

  }
// end of file
```