Policy-based Resolution of the Diffie-Hellman Protocol Vulnerability

El Hamzaoui Mustapha, Abderrahim Sekkaki, and Bahloul Bensassi

Department of Mathematics & Computer science
University Hassan II Aïn-chok, Faculty of sciences
P.O Box 5366, Maarif – Casablanca. Morocco
{m_elhamzaoui, a_sekkaki, bahloul_bensassi}@yahoo.fr

Abstract. Diffie and Hellman invented in 1976 a protocol bearing their names (Diffie-Hellman) that was at the origin of the cryptography. The most important characteristic of this protocol is that it enables two connected parties to generate a shared secret without having any preliminary information about one other. However, The point of vulnerability of this protocol is the active attack and more precisely the interception of the security information exchanges. The objective of this work is to solve the Diffie-Hellman vulnerability problem

1. Introduction

Cryptography plays, nowadays, a significant role in the realization of the security services and mechanisms (ISO 7498-2). Thus, several cryptography algorithms were developed in order to be used in security protocols and applications.

The cryptography algorithms require the use of the private and public keys. Moreover, in the case of distributed systems the management of the keys through a public keys infrastructures will be essential.

The Diffie-Hellman protocol [4] was the origin of public-key cryptology but its vulnerability point is the active attacks. Several approaches were proposed to circumvent the interception problem but they continue to exchange the security information between the connected parties and also they eliminate the possibility of generating a shared secret without having any preliminary information.

To avoid any security information exchange between connected parties, we will use an environment of security dynamic management to deal with all security tasks such as the public values exchanges, the private values generation, the shared secrets calculation, the keys' generation etc.

Policy-based management first consist in determining the strategies and the tactics reflecting the managers' objective and also representing them in policies' form. To distribute and apply these policies, they must be communicated to a Policy Decision Point (PDP) and to the Policy Enforcement Points (PEPs) managed by this PDP.

2. Outline on Diffie-Hellman protocol vulnerability

The Diffie-Hellman protocol [4] enables two connected parties to generate a shared secret without having any preliminary information about one another. However the Diffie-Hellman vulnerability is the interception (man-in-the-middle attack) of the exchanged security information. The principle of Diffie-Hellman protocol interception is:

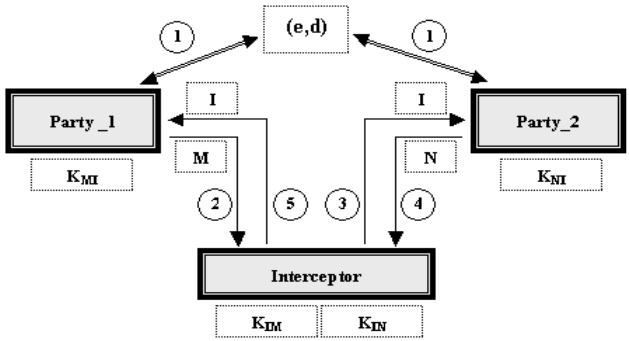

Fig. 1 The two connected parties have the impression to have the same shared secret (K_{MN}=K_{NM}). However, the interceptor shares with the party_1 the secret K_{MI}=K_{IM}, and he also shares with the party_2 the secret K_{NI}=K_{IN} .

1°- the connected parties define initially their integer public values (e,d).
2°- party_1 chooses a secret value m to calculate the public value M ($M=d^m \bmod e$) that will send after to party_2. The interceptor receives this value and will calculate, by using his internal secret value i, the corresponding shared secret: $K_{IM} = M^i \bmod e$.
3°- the interceptor calculates the public value I ($I=d^i \bmod e$) and will send it afterwards party_2 instead of the intercepted value M. At the reception of I, the party_2 will calculate the corresponding shared secret: $K_{NI} = I^n \bmod e$.
4°- the party_2 sends his value N ($N=d^n \bmod e$) to party_1. The interceptor will intercept this value to calculate the corresponding shared secret: $K_{IN} = N^i \bmod e$.
5°- the interceptor replaces N by I then sends it to party_1. Party_1 will calculate the corresponding shared secret: $K_{MI} = I^m \bmod e$.

Henceforth, we will call Diffie-Hellman parameters all values e, d, M, N, m, n, K_{MN} and K_{NM}.

3. The Ponder Policy language

Ponder [2] is an object-oriented, declarative language for specifying security and management policies for distributed system. Like any object-oriented languages, Ponder provides reuse by supporting types definition, which can be instantiated for

each specific situation by passing necessary parameters. Access control specification, obligation policy specification, constraints specification and composite policies specification are the basic characteristics of policy Ponder language.

Ponder is used in many works. Lymberopoulos et al. showed, in [8], how PONDER policies can be implemented and validated for Differentiated Services (DiffSer) by using CIM (Common Information Model) as the modeling framework for network resources as this device independent. They also used, in [9], Ponder language to realize a dynamic adaptation of policies in response to changes could occur within the managed environment. Finally, Damianou et al. presented in [3] the implementation of an integrated toolkit for the specification, deployment and management of policies specified in the PONDER language.

4. Our contribution to the solution of the Diffie-Hellman vulnerability problem

The main idea of our approach is to use a Keys' dynamic management environment to circulate no encoding/decoding information between connected parties in order to prohibit all interceptions between them.

In order to realize a dynamic management environment, two policy-based management levels will be defined. The first level will relate to the Diffie-Hellman parameters management while the second one, called high level, will deal with the management of the Diffie-Hellman parameters management policies themselves.

4.1. Policies of Diffie-Hellman parameters management

The plate-form to manage Diffie-Hellman parameters is:

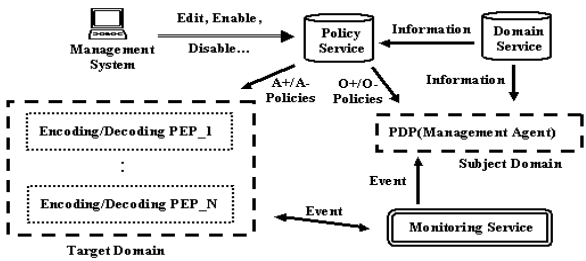

Fig.2. The Diffie-Hellman parameters management policies are triggered through the event EvtparamConf() which is activated automatically after the expiry of the estimated application period of the applied Diffie-Hellman parameters.

A Keys' management policy must be able to generate, distribute, store and remove automatically the keys just after the reception of a trigger-event :

```
inst oblig KeyMgmt11{//PPMA: Policy Parameters Mgmt Agt
```

```
on      EvtparamConf() ;
Subject resources/Soft/Agent/AgtPPMA/DHParAgt1;
Target  tgt=resources/Devices/PEP_Diff_Hellman/Dom1;
do      p[]=selp(genKeypub(),genKeypr1(),genKeypr2())
        -> Scr=calsecret(p[])
        -> EvtPolicyTrg(p[],Sec)||t.cfdev1(p[],Sec);
```

With the Reception of the event EvtparamConf(), the subject DHParAgt1 must perform, on the level of the target tgt, the following management actions where the 'll' concurrency operator allows the actions to perform in parallel while the '–>' is used to separate a sequence of actions:

Reference to a parameters' selection method selp() to choose randomly the Diffie-Hellman parameters (e,d), (m,M) and (n,N) (paragraph 2) respectively through the methods genKeypub(), genKeypriv1() and genKeypriv2(). Afterwards, the shared secret will be calculated by calling the method calsecret(). Thereafter, the event EvtPolicyTrg() will be activated to trigger the policy KeyMgmt12 that manages automatically the PEPs of the domain Dom2 wishing to apply the Diffie-Hellman protocol to communicate with the PEPs of the target domain tgt:

```
inst oblig KeyMgmt12 {

   on       EvtPolicytrigger(par1[],par2) ;
   Subject  resources/Soft/Agent/AgtPPMA/DHParAgt2;
   Target   t=resources/Devices/PEP_DH/Dom2;
    do      t.cfdev2(par1[],par2);
```

In parallel to this last action, the shared secret and the other calculated parameters will be transmitted to the methods cfdev1() existing on the level of the PEPs of the target domain tgt to configure them.

4.2. Policies to manage Diffie-Hellman parameters management policies

In order not to use durably the same security policy and give luck to others to discover our security parameters, we will use dynamic management policies to mange the Diffie-Hellman parameters management policies:

```
inst oblig PolicyMgmt {//PMA: Policy Management Agent

  on       EvtPolicySelect() ;
  Subject  resources/Soft/Agent/AgtPMA/DHPolMgtAgt;
  Target   t = /Policies/DH_Policy;
   do      Hpolicy= selectPolicy() -> Hpolicy.enable()
           Hpolicypar[] = selectPolicypar(Hpolicy)->
           GenerateEvt(PolicyObligEvt, Hpolicypar[]);
```

If the estimated time of the application of the current applied policy expires the event EvtPolicySelect() will trigger automatically the policy PolicyMgmt:

The subject DHPolMgtAgt selects firstly, by using the method selectPolicy(), the key management policy to apply and activates it. Then, the parameters corresponding to this policy will be calculated through the method selectPolicypar(). Finally, the

event GenerateEvt will be sent to transmit the parameters and to also trigger the selected policy through the obligation event PolicyObligEvt.

4.3. Role-based access control to secure resources

The notion of the role is used in several works on management such as distributed systems management [7], access control management [5] or virtual organization management [6]. The role is strongly related to the concept of position which is primarily a static concept describing a statute in an organization''

Access control to the users' PEPs. The users' PEPs which are the exchanged data encoding/decoding tools that apply the security policy decided by the domain PDP. Because of this importance, the PEPs access must be role-based controlled:

```
type role PEPs_Mger(set tgt) {
      type auth+ PEPAccessCtrl(target tgt1){
        action load(), remove(), enable(), disable();}
      inst auth+ PEP_DH_CtrlAcces = PEPAccessCtrl(tgt)
      type oblig PEP_Supervisor(target tgt2) {
         on  EventFail() ;
         do  intervene(tgt2); }
      inst oblig PEP_DH_Supervisor=PEP_Contoller(tgt);
... //other basic\composite policies of the role.
}// End of the role type declaration.
//  Domains specification:
Domain dmPerson=Personal/Admin/Admin_Diff_Hellman;
Domain P_DH=resources/Device/PEP/PEP_Diff_Hellman;
//Instantiation of the role PEPs_Manager:
inst role PEP_Dom1_Mger=PEPs_Mger(P_DH/Dom1) @dmPerson;
inst role PEP_Dom2_Mger=PEPs_Mger(P_DH/Dom2)@dmPerson;
```

Access control to the policies parameters management agents and policies management agents. In the same way, a role type must be specified for the management of agents that are indicated with the policies parameters management and the management of the policies themselves.

5. Evaluation of our work

The several approaches which were proposed to circumvent the interceptor attack problem such as SKIP protocol (Simple Key management for Internet Protocols) [1] and Photuris protocol [10] were all based on the authentication of the public values that will be used to generate the shared secret.

All approaches based on the Diffie-Hellman principle eliminate a significant property of Diffie-Hellman protocol which is the possibility of generating a shared secret. On the contrary, our solution preserved this characteristic and avoids the exchange of information security which is at the origin of all attacks and threats.

Moreover, our solution replaces the security information exchange by an automatic management based on the methods invocation that work on behalf connected parties.

6. Conclusion

The objective of our work was the presentation of an approach to solve the Diffie-Hellman vulnerability problem by preventing the interceptions between the connected parties. Our approach is based on the idea not to leave any encoding/decoding information circulate between connected parties and also to treat all actions that the connected parties could do on Diffie-Hellman parameters on the level of a Keys' management environment. This environment decides and applies, in an intelligent and dynamic way, the Diffie-Hellman parameters and the suitable security policies.

7. References

1. Ashar, A., Markson, T., Prafullchandra, H.: Simple Key-Management For Internet Protocols (SKIP). Internet Draft:draft-ietf-ipsec-skip-07
2. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The Ponder Policy Specification language. Proc. Policy 2001, International Workshop on Policies for Distributed Systems and Networks, Bristol, United Kingdom, January (2001) 29-31
3. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: Tools for Domain-Based Management of Distributed Systems. IEEE/IFIP Network operations and management symposium (NOMS2002), Florence,Italy,15-19 April (2002) pp.213-218
4. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory, IT-22, (1976) 644-654
5. Lupu, E.C, Marriott, D.A., Sloman, M.S., Yialelis, N.:A Policy Based Role Framework For Access Control., First ACM/NIST Role Based Access Control workshop, Gaithersburg,USA December (1995)
6. Lupu, E.C., Milosevic, Z., Sloman, M.S.: Use of Roles and Policies for Specifying, and Managing a Virtual Enterprise. Proceedings of the 9th IEEE International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE-VE'99). Sydney, Australia, March 23-24 (1999)
7. Lupu, E.C., Sloman, M.S.: Towards A Role Based Framework For Distributed Systems Management. Journal of Network and Systems Management, vol.5, no.1, Plenum Pess,. Systems Management, ,Plenum Press (1997) 2(4):333-360
8. Lymberopoulos, L., Lupu, E., Sloman, M.: PONDER Policy Implementation and Validation in a CIM and Differentiated Services Framework. 9th IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, Korea, May (2004)
9. Lymberopoulos, L., Lupu, E.,Sloman, M.: An Adaptive Policy-Based Framework for Network Services Management. Journal of Network and Systems Management, Vol.11, No.3, September (2003)
10. Simpson, W.A., Phil, K.: Photuris: Session-Key Management Protocol. RFC 2522, Mars (1999)