Dynamic Management of IPSec Security Policies Distribution

Abderrahim Sekkaki and El Hamzaoui Mustapha

University Hassan II Aïn-chok, Faculty of Sciences
Department of Mathematics & Computer Science
P.O Box 5366, Maarif – Casablanca. Morocco

Summary

Because of different constraints such as the customers unceasing requirements, the large distribution of systems, the permanent modifications of the management environments themselves, etc., the environments devoted to the management of the inter-domain communications security must be generally dynamic and policy based. In this work, we will present a management environment, that is mainly based on a dynamic platform, to policy-based manage the inter-domain communications security. Our proposed platform uses IPSec protocol and is composed of a security IPSec Server (IPSecServ) to decide and perform all management tasks, a Monitoring Service (MS) to automate the IPSecServ functioning, and a Policy Decision Point (PDP) with a set of Policy Enforcement Points (PEPs) to decide and distribute IPSec security policies. Moreover, our proposed approach integrates also domains, roles, and policies specification language PONDER to organise the environment components and facilitate their management. A prototype is implemented by using CORBA environment and some experimental results are also presented.

Key words :
Domain, IPSec, Policy-based management, Ponder, Role.

1. Introduction

The management of the inter-domain communications security requires the use of dynamic and policy based environment to overcome a set of problems like the big numbr of enterprises to manage and the permanent modifications that could occur inside the management environments themselves.
The main objective of the policy based management is the optimisation of administrator efforts and the automatisation of management. To specify policies, Ponder language [1] is actually an important tool to specify both security and management policies for distributed systems.

Our approach will be based on a dynamic platform to manage the IPSec security policies assuring the inter-domain communications security. The basic elements of this platform will be IPSec protocol [2], Ponder Language, roles [3], and domains [4] to facilitate the management of different elements of our security management environment (persons, materials, software, etc.).

A domain is very similar to a directory or folder on a personal computer, and it is used to partition large systems according to some precise criteria. Domains make the management of distributed systems very easy and flexible and give the possibility to modify the domains' components without altering management policies.

IPSec protocol offers the necessary mechanisms to construct virtual private networks (VPNs) that permit to create, however the used network, secured tunnels between connected parties. The IPSec protocol security services are provided, in transport or tunnel mode, through IPsec extensions; Authentication Header (AH) [5] and Encapsulating Security Payload (ESP) [6].

IPSec protocol employs Security Association (SA) [7] to facilitate the management of the parameters used by its extensions AH and ESP (algorithms, keys, etc). Each SA is identified by three parameters that are the destination address, the identifier of the used IPSec extension (ESP or AH), and the Security Parameter Index (SPI). Since the SAs are one-way connection we have to define two SAs (a SA in each direction) in order to protect the two directions of a traditional communication.

This work belongs to the set of research works realised inside our group and devoted to the VLABs security management [8], accounting management [9], inter-domain communications [10], and resolution of the Diffie-Hellman protocol vulnerability [11]. Moreover, it is considered as a continuity of the work [12] that is devoted to the policy based management of the IPSec Security policies distribution.

This work will be presented as follow, in the second section we will present briefly the policy based management principle. The third section will display Ponder language while our proposed approach will be the subject of the fourth section. The fifth section will be reserved to related works. Finally, the conclusion will be featured in the last section.

2. Principle of policy-based management

The objective of the policy-based management is the optimisation as well as possible of administrators efforts. Thus, it first consists in determining the strategies and the

Manuscript received May 5, 2007
Manuscript revised May 20, 2007

tactics reflecting the managers' objective and also representing them in policies' form. Then, these policies must be presented as a set of rules to be understood by the management entities and stored in a Policy Repository (PR). The distribution and the application of these policies require to have these rules communicated to a PDP (Policy Decision Point) and to PEPs (Policy Enforcement Point) managed by this latter [13].

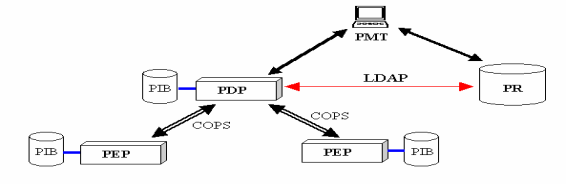

Fig.1. Policy-based management Platform

Concerning this management platform (fig.1), both a Policy Management Tool (PMT) and a PR must be placed on the higher level to allow administrator to configure the application level policies and store them afterwards in the PR (network policies level). The policies stored in the PEP PIB (Policy Information Bases) are called equipment level policies while those stored in the PDP PIB are called network level policies.

A second terminology was employed in the works devoted to management inside Imperial College [14]. This terminology uses the notions of Subject and Target instead of PDP and PEP. On one hand, Subjects indicate the manager objects and on other hand, Targets indicate the managed objects. The relations between Subject and Target are well defined by management policies and depend also on the nature of these latter. Thus, obligation policies define what a Subject must perform or not on the level of a Target, whereas authorisation policies specify the access rights that could have a Subject on the level of a Target.

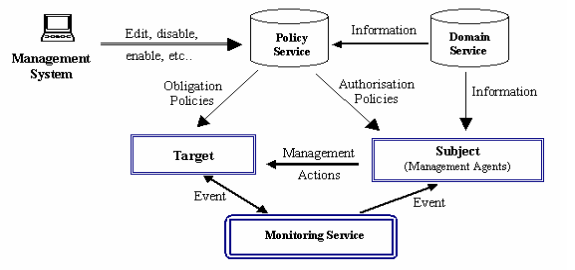

Fig.2. Management Platform based on the notions of subject and target.

PDP/PEP and Subject/Target are not contradictory notions but they are complementary. Thus, the PDP could play the Subject role and in the same way the PEP could play the Target role. The management platforms basing on the Subject and Target notions (fig.2) use generally a Monitoring Service to automate the management. Moreover, the policies rules database is replaced by both a domain service and a policy service. Concerning our work, the policies specification will be based on the Ponder language.

3. Policy Specification language

Ponder 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.

The Ponder basic characteristics are:

- Access control specification which is based on the deployment of authorisation, delegation, information filtering and refrain policies.
- Obligation policy specification to call upon managers to intervene when a special event occurs in the system.
- Constraints specification to define the conditions under which the policy is valid.
- Composite policies specification to simplify the policy specification task for large distributed systems.

For Ponder, subjects, targets, and policies are all organised in domains. The organisation in domains of the components of our security management environment is illustrated like this:

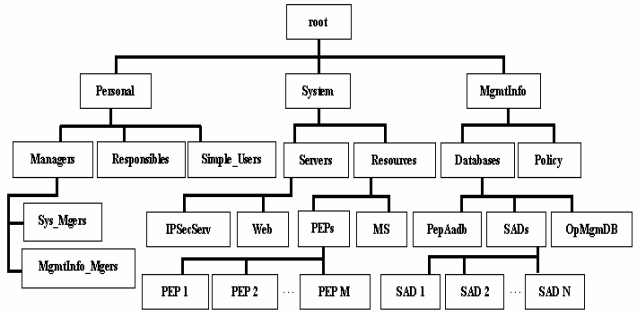

Figure 3. Organisation in domains of our management environment.

The dynamic management of the components of the security management environment such as users, materials,

software, etc.. requires, as it is schematised in fig.3, the use of a root domain. This latter organises these components in three main sub-domains; personal, System and MgmtInfo.

In what follows, we will restrict our discussion on obligation policies and roles.

3.1. Obligation policies

Obligation policies specify what activities a subject (members of one or several domains) must do to a set of target objects (objects of one or several domains) and define the duties of the policy subject. Obligation policy are triggered by events and are normally interpreted by a manager agent at the subject. Ponder defines two syntaxes to specify obligation policies:

- Syntax for direct declaration of obligation policy instance:

```
inst oblig policyName "{"
        on                      event-specification ;
        Subject     [<type>]    domain-Scope-Expression ;
        [ Target    [<type>]    domain-Scope-Expression ; ]
        do          obligation-action-list ;
        [ catch     exception-specification ; ]
        [ when      constraint-Expression ; ]   ''}''
```

The key word on specifies the required event. Subject and target are expressed in term of domains. The optional catch-clause specifies an exception that is executed if the actions fail to execute for some reason.

- Syntax for declaration and instantiation of authorization policy type:

```
Type oblig policyType ''('' formalParameters '')''
"{" {obligationation–policy-parts } ''}''

inst oblig policyName= policyType ''('' actualParameters '')'' ;
```

The obligation policy type is initially declared, then instantiated.

3.2. Role

The notion of the role is used in several works on management such as distributed systems management [15][16], access control management [17][18] or virtual organization management [19]. The role is strongly related to the concept of position and it has several definitions which converge toward the following idea: ''the role indicates, in the form of policies, the actions of management (rights and duties) representing the behaviour or the dynamic aspects of the position which is primarily a static concept describing a statute in an organization''

- The syntax to specify a role is:

```
inst role roleName "{"
        { basic–policy-definion }
        { group-definition }
        { meta-policy-definition } ''}'' [ @ subject-domain ]
```

A role can contain a certain number of basic policies, groups and meta-policies. The subject domain of the role can be optionally specified after the sign ' @ ' and if it is not specified a subject domain with the same name will be created by default.

Ponder was used in many research works. Thus, Lymberopoulos et al. showed, in [20], 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 [21], 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 [22] the implementation of an integrated toolkit for the specification, deployment and management of policies specified in the PONDER language.

In this work, we aim to automate as well as possible the security management. Dynamic management is actually a great need for large enterprises because the traditional solutions resting on the managers physical displacement became insufficient and very exceeded. Our proposed solution will be based on some important concepts such as policy based management integrating intelligent agents, Ponder language, roles and domains.

4. Our approach

4.1. Principle of our proposed approach

Our approach is devoted to the management of the inter-domain communication security. Precisely, it deals with the dynamic management of the IPSec security policies distribution.

Concerning the managed domains, each domain contains a set of PEPs. A part of these latter manages the intra-domain communications security whereas the other part manages the inter-domain communications security. To secure the inter-domain communications our attention will be concentrated solely on the PEPs assuring the inter-domain communications security (boundary PEPs):

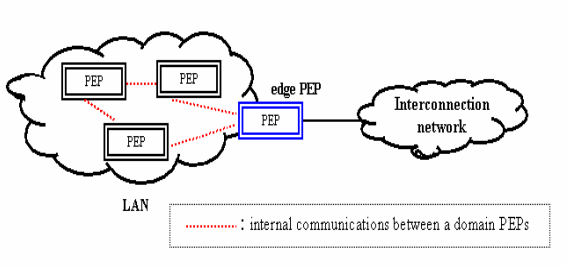

Figure 4. PEPs of a managed dmain (LAN).

In the context of policy based management, domains allow to organize the managed components and facilitate also their management while platforms automate the management itself. The possible interactions that could take place between the components of our security management environment are illustrated in figure 5 :

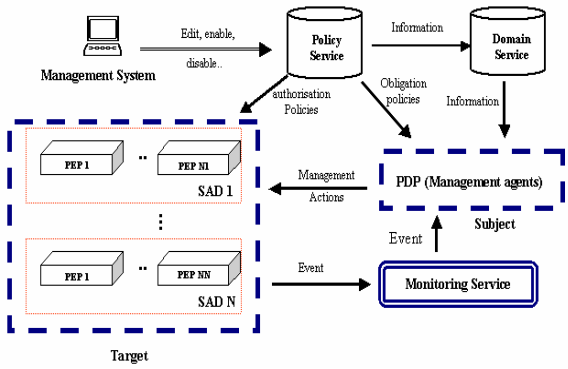

Figure 5. The proposed environment to manage the inter-domain communications security.

Policy and domain services are essential for the PDP to decide and distribute the IPSec security policies to apply on the level of its PEPs. Concerning the Monitoring Service integrated in this platform, it automates the PDP functioning.

The PDP is provided with a set of Security Association Databases (SADs) (fig.3) where each SAD contains three static tables; SA_param, Networks_Info and PEPs_Needs. The table SA_param contains ten SAs (SA_id, IPSec Extension, algorithm, ciphering key, IPSec mode, SPI) while the table Networks_Info contains the necessary PEPs IP addresses and networks masks. Concerning the table PEPs_Needs, it contains the SAs, selected and decided by the PDP, in order to put them in the disposal of its PEPs.

4.2. IPSecServ functioning

In order not to give luck to others to discover our security parameters, we must not use durably the same security parameters. Thus, we estimated two periods for changing the applied SAs and modifying the SAD static tables contents. Let T_{SA} and T_{SAD} be the periods of renewing the applied SAs and SADs databases contents. Precisely, to assure a good security level, we have to limit the number of changes of the PEPs IPSec configurations during each change of the SADs tables contents. Thus, we minimise the period T_{SAD} to prohibit any attempt to discover our security parameter and chose $T_{SAD} = 5 * T_{SA}$. Practically, the procedure is:

The MS asks, during each TSA, the PDP to change the SA applied by its PEPs. Moreover it asks also, during each T_{SAD}, the PDP to change the SADs contents. These two operations are realized by CORBA environment and are in the form of remote methods invocations as it is described on the interface (PkiServ_MS) of the following idl file (IPSec_PKI.idl):

```
module IPSEC_Serv {
  // Interface of methods invoked by the Monitoring Service:
   interface PkiServ_MS {
         oneway void modifySADParam();
         oneway void changeappliedSA();
   };
  ..... // other interfaces.
};
```

The specification of the obligation policy permitting to change the SAs, applied on the level of the PEPs, is :

```
inst oblig PolicyMgmtSA {
  on          EventChangeAppliedSA() ;
  Subject     System/Resources/Servers/IPSecServ ;
  Target      t = MgmtInfo/Databases/SADs ;
  do    id_SA = selectSAID() −> param_SA[] = selectParam(id_SA)
        −> storeSA(t.PEPs_Needs,param_SA[]) −> registryTask();
```

With the Reception of the event EventChangeAppliedSA(), the subject IPSecServ selects randomly, on the level of the table SA_param of each database of the target domain SADs (fig.3), a SA identifier and its corresponding parameters. The subject invokes afterwards the method StoreSA() to stores the resulted SAs in the table PEPs_Needs in order to put them in the disposal of its PEPs. Finally, the subject invoks the method registryTask() to registry this management task in a special database reserved for storing the subjects management operations (OpMgmtDB) (fig.3).

The corresponding UML modelisation is :

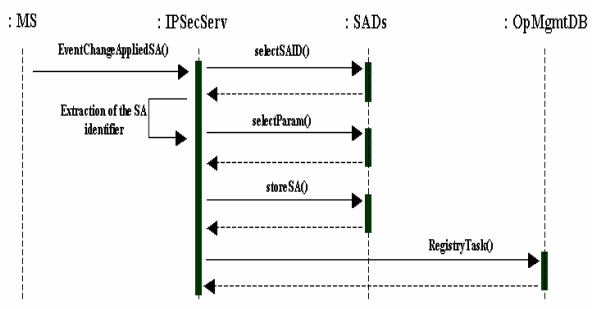

Fig.6. Diagram of sequence representing the necessary interactions to change the applied SA on the level of PEPs.

The specification of the obligation policy allowing to renew the SADs databases contents is:

```
inst oblig  PolicyMgmtSAD {
   on         EventChangeSADsContent() ;
   Subject    System/Resources/Servers/IPSecServ ;
   Target     t = MgmtInfo/Databases/ SADs ;
   do         delete(t) −> prtcl=Selectpro() −> algo=SelectAlgo(prtcl)
           −> key= calculKey(prtcl,algo) −> mode=SelectMode()
           −> spi=CalculSPI()
           −> StoreSAD(t.SA_param,prtcl,algo,key,mode,spi)
           −> registryTask();
```

With the reception of the event EventChangeSADsContent(), the subject IPSecServ proceeds in the following way to store ten new recordings in the tables SA_param of each database of the target domain SADs. Firstly, it calls the method delete() to erase the considered tables contents. Then, it calls the adequate methods to select and calculate the IPSec security parameters. Finally, the table SA_param of each database will receive ten new recordings through the method StoreSAD() and the operation will be registered in the database OpMgmtDB by invoking the method registryTask(). The corresponding UML modelisation is:

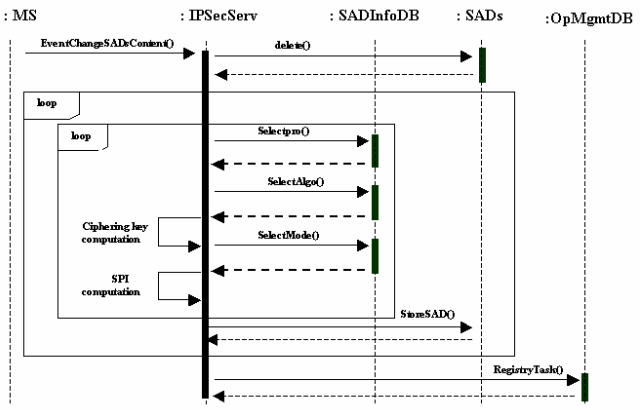

Fig.7. Diagram of sequence representing the necessary interactions to change the SADs' contents.

SADInfoDB is the main database used by the IPSecServ to fill SADs static tables (SA_param).

Concerning the PEPs, they consult periodically, during each T (T<TSA), the PDP to get a SA to apply. This task is developed by the CORBA environment and is assured precisely through the method getSAParam() of the interface PkiServ_PEPs of the following idl file IPSec_PKI.idl :

```
module IPSEC_Serv {
   ……
// Interface of methods invoked by the PEPs:
  interface PkiServ_PEPs {
    string getSAParam(in string pep_id, in string pep_passwd,SAD_id);
  };
};
```

In order to get the necessary SA to apply, the PEP has to send its own identity information (PEP identifier, PEP password, and PEP SAD identifier) to the IPSecServ. These information are received by the IPSecServ as arguments of the method getSAParam() to be used to identify it.

4.3. Role-based control of the resources access

The specification of roles allows to organise and facilitate resources and services management. Thus, the use of roles specifications in our approach will be usefull to protect our environment reesources and services from any unauthorized access.

- IPSecServ access control :

IPSecServ is the main component of our management environment. Thus, a role type must be specified to protect it against all possible dangers:

```
type role Servers_Manager (set tgt) {

/*Declaration and instantiation of an authorization policy type to control
the server access : */

        type auth+ ServersAccessCtrl (target tgt1){ action
configure(), remove(), enable(), disable(); }
        inst auth+ Serv_CtrlAcces = ServersAccessCtrl (tgt) ;

/* Declaration and instantiation of an obligation policy type to intervene
in the case of server breakdown : */

        type oblig Servers_Supervisor (target tgt1) {

                on Serv_FailEvent() ;
                do repair(tgt1); }
        inst oblig Serv_Controller = Servers_Supervisor (tgt);

        …….. \\ Other policies attributed to the role.
} // End of the role type declaration.

// Domains specification :

// System Managers Domain.

Domain dmnSysMgers = Personal/Managers/Sys_Mgers ;

// Security Server Domain.

Domain dmnServs = System/Servers/IPSecServ;

// Instantiation of the role Servers_Manager:

inst role role_Serv_Mger = Servers_Manager(dmnServs) @
dmnSysMgers;
```

- Databases access control :

Because of the sensibility of our environment databases information, a role type must be also specified to control their access :

```
type role DBs_Manager (set tgt) {

/*Declaration and instantiation of an authorization policy type to control
the access to the target : */

 type auth+ DBsAccessCtrl (target tgt1){
       action configure(), remove(), add(), modify(); }
 inst auth+ DB_CtrlAcces = DBsAccessCtrl (tgt) ;

/* Declaration and instantiation of an obligation policy type to intervene
in the case of the target breakdown : */

        type oblig DBs_Supervisor (target tgt2) {

                on DB_FailEvent() ;
                do intervene(tgt2); }
        inst oblig DB_Controller = DBs_Supervisor (tgt);

        …….. \\ Other policies attributed to the role.
} // End of the role type declaration.

// Domains specification :

// System Managers Domain.

Domain dmnSysMgers = Personal/Managers/Sys_Mgers ;
// Databases Domain.
Domain dmnDBs = MgmtInfo/Databases;

// Instantiation of the role DBs_Manager:

inst role role_DBs_Mger = DBs_Manager(dmnDBs) @
dmnSysMgers;
```

- Domain 'Simple_Users' access control :

The domain Simple_Users contains the necessary information on the environment users. Therefore, it must be protected, through a role type specification, against all dangerous access such as unauthorized add/suppression of users, destruction, etc. :

```
type role Users_Manager (set tgt) {

/*Declaration and instantiation of an authorization policy type to control
the access to the target: */

        type auth+ Users_accessCtrl (target tgt1){
            action Add(), remove(),Modify(); }
        inst auth+ Users_ACCtrl = Users_accessCtrl (tgt) ;

/*Declaration and instantiation of an obligation policy type to intervene
in the case of the target dysfunction : */

      type oblig Users_Supervisor (target tgt2) {

                on Users_FailEvent() ;
                do intervene(tgt2); }
       inst oblig Users_Controller = Users_PEPs_Supervisor (tgt);

        …….. \\ Other policies attributed to the role.
} // End of the role type declaration.

// Domains specification :

// System Managers Domain.
Domain dmnSysMgers = Personal/Managers/Sys_Mgers ;
Domain dmnUsers = Personal/Simple_Users; // PEPs Domain.

// Instantiation of the role Users_Manager:

inst role Users_Mger = Users_Manager(dmnUsers) @ dmnSysMgers;
```

- PEPs access control :

A role type must be also specified to manage the PEPs of our environment against all dangerous access (suppression/add of PEPs to the sub-domains of the domain PEPs (fig.4)):

```
type role PEPs_Manager (set tgt) {

/*Declaration and instantiation of an authorization policy type to control
the access to the target : */

        type auth+ PEPsAccessCtrl (target tgt1){
              action Add(), remove(),Modify(); }
        inst auth+ PEPs_CtrlAcces = PEPsAccessCtrl (tgt) ;

/*Declaration and instantiation of an obligation policy type to intervene
in the case of the target dysfunction : */

        type oblig PEPs_Supervisor (target tgt2) {

                on PEPs_FailEvent() ;
                do intervene(tgt2); }

        inst oblig PEPs_Controller = PEPDom_Supervisor (tgt);

        …….. \\ Other policies attributed to the role.
} // End of the role type declaration.

// Specification of domains :
```

```
// System Managers Domain..
Domain  dmnSysMgers = Personal/Managers/Sys_Mgers ;
Domain  dmnPEPs = System/Resources/PEPs;   //Domain 'PEPs'.

// Instantiation of the role PEPs_Manager :

inst role  role_PEPs_Mger = PEPs_Manager(dmnPEPs)  @
dmnSysMgers;
```

4.4. Results and execution

In our example, the IP addresses of our environment sub-networks (fig.8) are of the form 11.0.j.0 (1<j<255) and the address of the interconnecting network is 50.0.0.0. The IP addresses of the PEPs assuring the inter-domain communications security are illustrated like this:

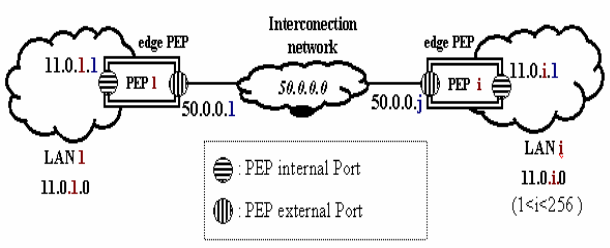

Figure 8. IP adressing

In order to simplify the implementation of our prototype, we used three databases; two databases SADs (SAD1.mdb and SAD2.mdb) and a database PepAadb.mdb (PEP Authentication and Authorization Database) (fig.3). The database PepAadb.mdb contains only a static table (PEP_Info) (fig.9).

PEP_Info : Table

PEP_id	password	SAD	destination_PEP_id
1	PEP1_pw	SAD1	6
5	PEP5_pw	SAD2	9
6	PEP6_pw	SAD1	1
9	PEP9_pw	SAD2	5
10	PEP10_pw	SAD1	15
15	PEP15_pw	SAD1	10

Enr : 7 sur 7

Fig.9. Content of the table PEP_Info of the database PepAadb.mdb.

The next executions will be restricted to the databases SAD1.mdb (tables SA_param, Networks_Info and PEP_Needs) (fig.10), and PepAadb.mdb (fig.9).

SA_param : Table

SA_id	IPSec_Ext	Algorithm	Chipher_Key	IPSec_Mode	SPI
0	AH	blowfish-cbc	tatatatatata	tunnel	3000
1	ESP	hmac-sha1	aaaabbbbaaaabbbbaaaa	transport	1500
2	ESP	3des-cbc	rajarajarajarajarajaraja	tunnel	1200
3	AH	des-cbc	authrauthz	tunnel	2500
4	ESP	hmac-md5	rca2006rca2006rc	transport	1500
5	AH	hmac-sha2-256	qiitqiiitqiitqiitqiitqiitqiit	transport	3002
6	AH	keyed-sha1	rcarcarcarcarcarcarc	tunnel	3621
7	AH	hmac-md5	tictactictactict	tunnel	1300
8	ESP	rijndael-cbc	titotitotitotito	tunnel	2505
9	AH	aes-xcbc-mac	groupedetravailg	transport	3666

Enr : 1 sur 10

PEPs_Needs : Table

Direction	IPSec_Ext	Algo	Key	IPSec_Mode	SPI
1	ESP	hmac-sha1	aaaabbbbaaaabbbbaaaa	transport	1500
2	AH	keyed-sha1	rcarcarcarcarcarcarc	tunnel	3621

Enr : 1 sur 2

Networks_Info : Table

PEP_id	PEP_IP_adr	Net_Mask
1	50.0.0.1	255.255.255.0
3	50.0.0.3	255.255.255.0
6	50.0.0.6	255.255.255.0
9	50.0.0.9	255.255.255.0
10	50.0.0.10	255.255.255.0
15	50.0.0.15	255.255.255.0

Enr : 1 sur 6

Figure 10. Tables of the databases SAD1.mdb.

Concerning the renew of the PEPs IPSec configurations and SADs tables contents, we chose, as we already mentioned, to change five times the PEPs IPSec configurations during each renew of the SADs tables contents. Thus, the method changeSA() is invoked five successive times during each invocation of the method changeSADcontent().

Concerning the execution of the prototype, the content of the table SA_param (fig.9) was the result of the second invocation of the method changeSADcontent(). With these table SA_param data, the method changeSA() was invoked five successive times and the content of the table PEPs_Needs (fig.9) corresponds to the third invocation of this method.

The implementation the methods changeSADcontent() and changeSA(), on the level of the PKIServ, is:

```
…… //  Program
public String  changeSADContent(){
  // A loop to act on the databases
   for (int i=1; i<=2; i++) { SAD1.mdb and SAD2.mdb.
// Method to suppress  the contents of the considered databases tables.
      delete(SAD[i]);
 // Method to change the contents of the selected databases.
 changeContenent(SAD[i]);
  }
}
…… // Program
public String  changeSA(){
```

```
// A loop to act on the databases SAD1.mdb and SAD2.mdb.
  for (int j =1; j<=2; i++) {
      int r = (int)(Math.random()*10);
      int  id_SA = r % 3;  // Random choice of a SA identifier.
      // Selection of  the considered SA parameters .
      String param = selectSAParam(SAD[j],id_SA);
      // Method to store the SA selected in  the table PEPs_Needs.
      Store(SAD[j],param); }
}
…… // Program
```

For this content of the table PEPs_Needs (fig.9), we dealt with the example of the PEPs pair (PEP1,PEP6) assuring the security of the communications between the domains LAN1 and LAN2. Thus, in order to change its IPSec configuration, the PEP1 (PEP_id_source = 1) proceeded in the following way:

Firstly, it requested the IPSecServ, through the method getSAParam(), to get a SA to apply. Secondly, the PDP checked the PeP identity and authorisation through the consultation of database PepAadb.mdb (table PEP_Info (fig.9)). Finally, The IPSecServ gave it the following response:

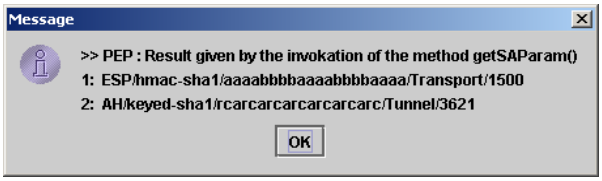

Fig.11. Obtained parameters to construct the desired SA

As we already discussed, a SA is one-way connection. Consequently, two SAs (a SA in each direction) must be defined to protect the two directions of a traditional communication.

Concerning our execution, the Obtained SA is composed of two sub-SAs (two lines) and concerns two PEPs. The SA lines order depends on the position of the PEPs in the table PEP_Info (PepAadb.mdb (fig.9)). Thus, the PEP located the first in the PEP_Info receives the two lines in the order which they have in the table PEPs_Needs else it will receive them in the reversed order.

Consequently, with the reception of the desired SA (fig.11), the PEP1 changed its IPSec configuration and its new IPSec configuration was then:

Add 50.0.0.1 50.0.0.6 esp 1500 –m transprot –A hmac-sha1 ‘‘aaaabbbbaaaabbbbaaaa ’’

Add 50.0.0.6 50.0.0.1 ah 3621–m tunnel –E keyed—sha1 ‘‘rcarcarcarcarcarc ’’

The PEP6 proceeded also in the same way to renew its IPSec configuration. Its new IPSec configuration was:

Add 50.0.0.6 50.0.0.1 ah 3621–m tunnel –E keyed—sha1 ‘‘rcarcarcarcarcarc ’’

Add 50.0.0.1 50.0.0.6 esp 1500 –m transprot –A hmac-sha1 ‘‘aaaabbbbaaaabbbbaaaa ’’

5. Related works

IPSec protocol was the objective of several research works. Theses works have either studied the protocol IPSec itself or used it to develop some security solutions.

In the contexte of security solutions, an approach is presented in [23] to distribute the IPSec security policies. This approach is based on an infrastructure characterised by the use of PDP, PEPs, PIB, COPS-PR protocol, LDAP protocol and a database of policy rules.

In the same contexte, Al-Chaal has presented a dynamic and centralized approach [24] that is easily administrable and based on the VPN technology (IPVPNs: IP Virtual Private Networks). This approach is is based on a Network Operation System (NOS) to deal with all management tasks such as group adhesion, VPN topology creation, etc.. Moreover, this approach contributes also in web services security and techniques of load division and performances amelioration.

In the contexte of works devoted to the study of the IPSec protocol itself, many studies have been developped [25] to discuss the mechanisms and the main uses of the IPSec protocol.

Our proposed approach rests on the research works devoted to policy based management inside Imperial College. It is policy based and characterized by the employment of important concepts such as domains, Ponder language, roles, monitoring service, CORBA objects , PDP and PEPs. Concerning the CORBA environment, it was very useful in the implementation of the IPSecServ-MS and IPSecServ-PEPs communication thanks to remote methods invocation that provides. These basic elements permitted us to develop a dynamic, flexible and extensible platform to manage the inter-domain communications security. The flexibility and extensibility of policy-based management concept will allow to apply it in other architectures and security solutions such as distributed systems, distributed virtual laboratories, databases, Web servers, etc.

6. Conclusion

Because of the vulnerability of systems and the various attacks which could target enterprises such as spying, piracy and destruction, security became an important and decisive parameter which must be taken in consideration during each exchange of information between domains. Then, the inter domain communications security requires a special effort because the traditional solutions are now insufficient and exceeded. The current tendency is the dynamic security through intelligent environments characterized by their flexibility and extensibility.

Our work constitutes a contribution to the efforts devoted to this subject. Thus, we proposed a dynamic IPSec security infrastructure which decides and distributes IPSec security associations at exact times. Moreover, the proposed infrastructure allows to change, in an intelligent way, the contents of the security databases without any intervention of human.

Our security infrastructure is also characterized by the use of Ponder language, roles, and domains which permit to organize and facilitate more the management of the security environment.

7. Bibliographie

[01] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. "The Ponder Policy Specification language". Proc. Policy 2001, International Workshop on Policies for Distributed Systems and Networks, Bristol, United Kingdom, January

[02] S. Kent, and R. Atkinson. ''Security Architecture for the Internet Protocol'', RFC 2401, November 1998. 29-31 2001.

[03] M.S. Sloman. ''Policy Driven Management for Distributed Systems'', Journal of Network and Systems Management,2(4):333-360,Plenum Press, 1994.

[04] M.S. Sloman and K.P. Twidle. ''Domains: A Framework for Structuring Management Policy'', in Network and distributed systems management, ed. Morris Sloman, Addison-Wesley, pp.433-453, 1994.

[05] S. Kent, and R. Atkinson ''IP Authentication Header'', RFC 2402, November 1998.

[06] S. Kent, and R. Atkinson, ''IP Encapsulating Security Payload (ESP)'', RFC 2406, November 1998.

[07] hsc: http://www.hsc.fr/ressources/articles/ipsec-tech/, last update : 23 October 2002.

[08] A. Sekkaki, M. El Hamzaoui, B. Bensassi. ''Policy-based Management of a Virtual Laboratory Communications Security''. In Proc. The First IEEE International Workshop on Broadband Convergence Networks (BCN2006), pp.199-204, Vancouver, Canada, April 7 (2006).

[09] C.B. Westphall, A. Sekkaki, L.M. Alvarez and W.T. Watanabe. ''Extending TINA Secure On-Line Accounting Services'', Journal of Network and Systems Management, Vol.11, No.4, December 2003.

[10] M. El Hamzaoui, A. Sekkaki, B. Bensassi, ''Policy-Based Management of the inter-Domain communications Security'', in: Proc. IEEE/IFIP 4th Latin American Network Operations and Management Symposium (LANOMS), pp. 269-274, Porto Alegre, Brazil, 2005.

[11] M. El Hamzaoui, A. Sekkaki, B. Bensassi, ''Policy-based Resolution of the Diffie-Hellman protocol vulnerability'', in: Proc. IEEE/IFIP 4th Latin American Network Operations and Management Symposium (LANOMS), pp. 277-282, Porto Alegre, Brazil, 2005.

[12] M. El Hamzaoui , A. Sekkaki, B. Bensassi. '' Infrastructure to manage the IPSec Security Policies''. In Proc. GRES'2006 - Gestion de REseau et de Service, 6ème Colloque Francophone, Bourdeaux-France, 2006.

[13]. R. Yavatkar, D. Pendarakis, and R. Guerin. '' A Framework for Policy-based Admission Control '', RFC 2753, January (2000).

[14] Imperial College : http://www.doc.ic.ac.uk/, last update : 2003.

[15] E.C. Lupu. "A Role-Based Framework for Distributed Systems Management". PhD. Dissertation, Imperial College of Science, Technology and medicine, University of London, UK,1993.

[16] E.C. Lupu, and M.S. Sloman. ''Towards A Role Based Framework For Distributed Systems Management'', Journal of Network and Systems Management, vol.5, no.1, Plenum Pess. Systems Management,2(4):333-360,Plenum Press,1997.

[17] R. Croock, D. Ince, and B. Nuseibeh. ''Modelling access policies using roles in requirements engineering'', Information and Software Technology 45(2003)979-997, April 2003.

[18] E.C. Lupu, D.a. Marriott, M.S. Sloman and N. Yialelis. ''A Policy Based Role Framework For Access Control'', First ACM/NIST Role Based Access Control workshop, Gaithersburg, USA, December 1995.

[19] E.C. Lupu, Z. Milosevic, and M.S. Sloman. "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.

[20] L. Lymberopoulos, E. Lupu, and M.Sloman. "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.

[21] L. Lymberopoulos, E. Lupu, and M.Sloman. ''An Adaptive Policy-Based Framework for Network Services Management''. Journal of Network and Systems Management, Vol.11, No.3, September 2003.

[22] N. Damianou, N. Dulay, E.C. Lupu and M.S. Sloman. ''Tools for Domain-Based Management of Distributed Systems'', IEEE/IFIP Network operations and management symposium (NOMS2002), pp.213-218, Florence,Italy,15-19 April,2002.

[23] F. Barrère, A. Benzekri, F. Grasset, R. Laborde and Y. Raynaud. '' Distribution de politiques de sécurité IPsec'', GRES'01-Gestion de Réseau et de Service,4ème Colloque Francophone, pp. 271-284, Marrakech-Maroc, Décembre2001.

[24] L. Al-Chaal, "Dynamic and Easily Manageable Approach for Secure IP VPN Environments". Ph.D Dissertation, Institut National Polytechnique de Grenoble, France, 2005.
[25] TCP/IP Guide: http://www.tcpipguide.com/free/t_IPSecModesTransportandTunnel.htm, last update : September 2005.

Abderrahim Sekkaki received a D.Sc. in Network Management domain from the Paul Sabatier University, France, in 1991: and a Dr. of State Degree from Hassan II University, Morocco, in 2002. He does research on distributed systems and policies based network management. Presently, he is a Professor in Computer Science at the Hassan II University, Casablanca, Morocco.

El Hamzaoui Mustapha is currently a doctoral student in Faculty of Science, University Hassan II Aïn Chock. His primary research interest is in based-policy Management of networks security.