Demo User Guide

The main goal of the demo is to show how security policies are dynamically deployed using the Aspect Generation and the Aspect Weaver modules. The demonstration is organised in three different scenarios:

- **Scenario 1. Initial deployment of security aspects.** In this scenario three AspectJ aspects are initially deployed (Authentication, Encryption and ElapsedTimeMonitoring). The Authentication aspect implements the behaviour required to identify the client. The Encryption aspect is in charge of cyphering the client vote. Finally, the ElapsedTimeMonitoring aspect monitors the encryption process in order to calculate the time that it takes the vote cyphering. For the purpose of the demonstrator we assume that initially there is not any aspect deployed in the application. We will show how the list of aspects to be deployed is generated by the Aspect Generation module and how the aspects are deployed by the Aspect Weaver module.

- **Scenario 2. Redeployment/Adaptation of the behaviour of deployed aspects at runtime.** In this scenario the demonstration shows how aspects can be deployed/undeployed dynamically at runtime, and how the Aspect Generation module generates a security adaptation plan by calculating the 'difference' between a previous configuration and a new one, indicated by the new security policy. Concretely, the Encryption and the ElapsedTimeMonitoring aspects are un-deployed from the client and a new Encryption aspect is deployed in the server.

For the first and the second scenarios of the demonstration the aspects have been implemented in AspectJ and the pointcuts have been directly implemented as part of the aspect implementations. This is enough to show a proof of concept of the Aspect Generation and Aspect Weaver module, although in the next milestones it will be possible to implement more aspects in different AOP languages. Also, in next versions of the modules the pointcuts will be always specified separately from the aspect implementations in order to increase the reusability of the aspects. As an example of the implementation of the aspects in AspectJ, in the following figures you can see a simplified version of an authentication aspect, the pointcut to add the authentication aspect to the client side and the poincut to add the authentication aspect to the server side.

In Figure 1 we show how the aspects in INTER-TRUST will extend from the IntertrustAspect in order to incorporate common behavior that is required to deploy and/or undeploy the aspects in the INTER-TRUST framework. All of them will incorporate the mechanism that provides the Aspect Weaver module with the capacity of enabling and/or disabling them (see setEnabled() and isEnabled() methods). In this aspect the poincut has been implemented as an abstract pointcut. This means that the pointcut will be defined by each application reusing this abstract aspect.

```
public abstract aspect Authentication_X509certificate extends IntertrustAspect {

    // ENABLING/DISABLING
    private static boolean enabled = false;

    @Override
    public void setEnabled(boolean enabled) {
        this.enabled = enabled;
    }

    @Override
    public boolean isEnabled() {
        return enabled;
    }

    pointcut enabled(): if(enabled);

    pointcut p1(): enabled() && authPoints();

    abstract protected pointcut authPoints();

    before(): p1() {
        authenticate();
    }

    public void authenticate() {
        //System.out.println("Authenticating using x509 certificate...");
        writeConsole("Authenticating using x509 certificate...\n");
    }

}
```

Figure 1. Simplified implementation of an authentication aspect in AspectJ

Figure 2 shows a concrete authentication aspect that extends the abstract aspect shown in Figure 1 in order to define the pointcut that indicates how to weave the authentication aspect with the client.

```
public aspect Authentication_X509certificate_EVotingClient extends Authentication_X509certificate {

    protected pointcut authPoints(): execution(* evoting.core.client.EVotingInt.vote(..));
}
```

Figure 2. Pointcut definition to incorporate the authentication aspect to the client side

Figure 3 shows another concrete aspect with the definition of the poincut to weave the authentication aspect with the server.

```
public aspect Authentication_X509certificate_EVotingServer extends Authentication_X509certificate {

    protected pointcut authPoints(): execution(* evoting.core.server.EVotingServerInt.receiveVote(..));
}
```

Figure 3. Pointcut definition to incorporate the authentication aspect to the server side

Aspectual Security Knowledge

The aspectual security knowledge required for this demo is provided in the folders “clientFiles” and “serverFiles” described in the setup guide. These folder contains all the files with the necessary aspectual security knowledge. They are also available to be visualized through the demo graphical interfaces as explained in the user manual. There are two different files:

1. Security Deployment Specification (SDS) files: There is two file for the initial deployment (one for the client and one for the server) and two more for each scenario (one for the client and one for the server in each scenario). This file is the output of the Policy Interpreter module and is used to notify a request for security policy adaptation. Figure 4 shows an example of an SDS file for an scenario. Basically, this file contains the security concepts that have to be deployed in the base application. For instance, authentication, encryption and timing are the security concepts for this example scenario and these are grouped into different security categories (access-control, cryptography, and monitoring, respectively). Each security concept has a list of required security functionalities.
   The mapping between the security policies (specified in ORBAC) and this file is out of the scope of this demo.

```
<?xml version="1.0" encoding="UTF-8"?>

<sds:sds xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:sds='http://inter-trust.eu/schema/interpreter/sds'
   xsi:schemaLocation='http://inter-trust.eu/schema/interpreter/sds SDS.xsd'>
    <sds:deploy>
        <sds:category id="http://inter-trust.eu/security/access-control">
            <sds:securityConcept id="2" type="http://inter-trust.eu/security/authentication">
                <sds:target id="http://inter-trust.eu/csa/vehicle/authnmodule"/>
                <sds:functionality id="http://inter-trust.eu/security/authentication#certificate-authentication"/>
                <sds:functionality id="http://inter-trust.eu/security/authentication#x509certificate"/>
                <sds:configuration>
                    <sds:securityParameters>
                        <sds:parameter name="CA">C=ES, O=DIRECCIÓN GENERAL DE TRÁFICO, OU=ITS, CN=DGT.GOB.ES
                        </sds:parameter>
                    </sds:securityParameters>
                </sds:configuration>
            </sds:securityConcept>
        </sds:category>
        <sds:category id="http://inter-trust.eu/security/cryptography">
            <sds:securityConcept id="3" type="http://inter-trust.eu/security/encryption">
                <sds:target id="http://inter-trust.eu/csa/vehicle/cam-sender"/>
                <sds:functionality id="http://inter-trust.eu/security/confidentiality#encryption"/>
                <sds:functionality id="http://inter-trust.eu/security/secure-message-format#pkcs7"/>
                <sds:configuration>
                    <sds:securityParameters>
                        <sds:parameter name="Algorithm">DSA</sds:parameter>
                    </sds:securityParameters>
                </sds:configuration>
            </sds:securityConcept>
        </sds:category>
        <sds:category id="http://inter-trust.eu/security/monitoring">
            <sds:securityConcept id="7" type="http://inter-trust.eu/security/timing">
                <sds:target id="http://inter-trust.eu/csa/vehicle/cam-sender"/>
                <sds:functionality id="http://inter-trust.eu/monitoring/timing#executiontime"/>
            </sds:securityConcept>
        </sds:category>
    </sds:deploy>
    <sds:undeploy>
    </sds:undeploy>
```

Figure 4. Example of the SDS file.

2. Mapping between security functionalities and aspects: There are two mapping files (one for the client and one for the server) including all the mapping information required for the demo. These files contains the information of the initial available aspects and the security functionalities that each aspect provides. Figure 5 shows an example of this kind of file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sds:listAF xmlns:sds="http://inter-trust.eu/schema/interpreter/sds" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://inter-trust.eu/schema/interpreter/sds
SAK_Schema.xsd ">

<sds:aspect id="1" name="evoting.aspects.client.Authentication_UserPassword_EVotingClient">
    <sds:advice id="1" name="authenticate">
      <sds:functionality id="http://inter-trust.eu/security/authentication#userpass-authentication"/>
    </sds:advice>
  </sds:aspect>
  <sds:aspect id="2" name="evoting.aspects.client.Authentication_X509certificate_EVotingClient">
    <sds:advice id="1" name="authenticate">
        <sds:functionality id="http://inter-trust.eu/security/authentication#certificate-authentication"/>
        <sds:functionality id="http://inter-trust.eu/security/authentication#x509certificate"/>
    </sds:advice></sds:aspect>
  <sds:aspect id="3" name="evoting.aspects.client.Encryption_DSA_EVotingClient">
    <sds:advice id="1" name="encryption">
        <sds:functionality id="http://inter-trust.eu/security/confidentiality#encryption"/>
        <sds:functionality id="http://inter-trust.eu/security/secure-message-format#pkcs7"/>
    </sds:advice>
    <sds:advice id="2" name="decryption">
        <sds:functionality id="http://inter-trust.eu/security/confidentiality#decryption"/>
        <sds:functionality id="http://inter-trust.eu/security/secure-message-format#pkcs7"/>
    </sds:advice>
  </sds:aspect>
  <sds:aspect id="4" name="evoting.aspects.client.Encryption_RSA_EVotingClient">
    <sds:advice id="1" name="encryption">
        <sds:functionality id="http://inter-trust.eu/security/confidentiality#encryption"/>
        </sds:advice>
    <sds:advice id="2" name="decryption">
        <sds:functionality id="http://inter-trust.eu/security/confidentiality#encryption"/>
        </sds:advice>
        </sds:aspect>
  <sds:aspect id="5" name="evoting.aspects.client.ElapsedTimeMonitoring_EVotingClient">
    <sds:advice id="1" name="time">
        <sds:functionality id="http://inter-trust.eu/monitoring/timing#executiontime"/>
    </sds:advice>
    </sds:aspect>
</sds:listAF>
```

Figure 5. Example of mapping security functionalities and aspects.

User Manual:

We have organised this user manual in seven main parts. The steps that need to be followed to execute and test the demo are the following:

1. **Execute the eVoting server.**

When the eVoting server is executed, following the setup guide in section 3.3.3, the e-Voting Server graphical interface will be initiated (see Figure 6).

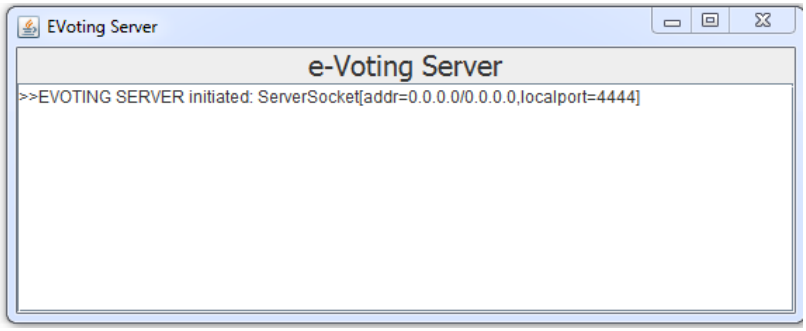

Figure 6. Graphical interface of the e-Voting server example prepared for the demo

Also, a graphical interface that will allow us to simulate the behaviour of the Policy Interpreter module is initiated (see Figure 7). In the INTER-TRUST framework, the Policy Interpreter module notifies to the Aspect Generation module the new security policies to

be deployed. Since it is not the purpose of the first milestone to demonstrate the Policy Interpreter module, we have temporarily implemented this code in our demonstrator in order to simulate the interactions of the Policy Interpreter module with the Aspect Generation module, with the purpose of better illustrating its behavior.

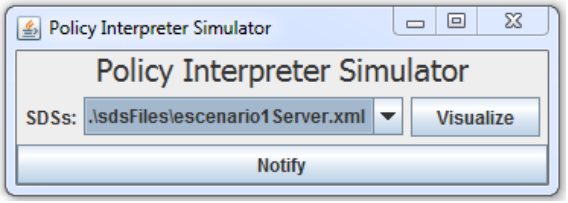

Figure 7. Graphical interface to simulate part of the behaviour of the Policy Interpreter module

The graphical interface in Figure 7 allows the user to:

a) *Select* the Security Deployment Specification (SDS) file to be deployed. This documents conform to the XML Schema specified as part of the specification and implementation of the Policy Interpreter module (see deliverable D2.3.1 for the description of this XML Schema).

b) *Visualize* the content of the selected SDS file.

c) *Notify* the SDS file to be deployed to the Aspect Generation module. Thus, with this notification we simulate the behaviour of the INTER-TRUST framework, where a negotiated security policy is interpreted by the Policy Interpreter module that generates a SDS file with a required adaptation. This adaptation request is notified to the Aspect Generation module by mean of the SDS file.

2. **Execute the eVoting client.**

When the eVoting client is executed, following the setup guide in section 3.3.3, the e-Voting Client graphical interface will be initiated (see Figure 8).

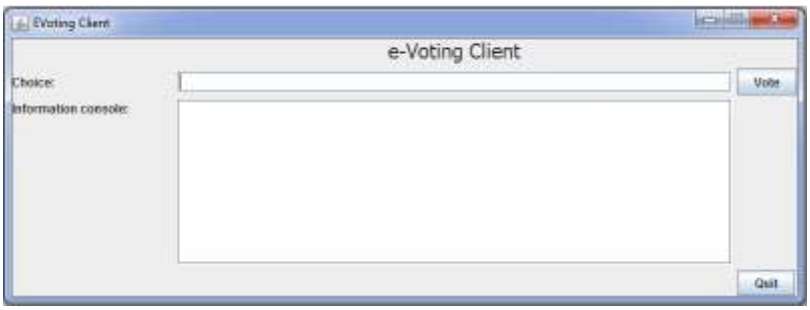

Figure 8. Graphical interface of the e-Voting client example prepared for the demo

Also, the Policy Interpreter Simulator is initiated (see Figure 7) at the client side. Notice that according to the INTER-TRUST software architecture an instance of the INTER-TRUST framework is initiated in every device participating in an INTER-TRUST application, and this is because we instantiate our modules (Policy Interpreter Simulator, Aspect Generation module and Aspect Weaver module) both at the client and at the server side of the e-Voting application.

3. **The client votes before the deployment of any aspect.**

For the purpose of the demonstrator, we assume that when the e-Voting application is initiated there is not any aspect deployed yet. This means that the e-Voting client can write its choice (e.g. option 1 in Figure 9), press the 'Vote' button and the vote will be received in the server.

We can observe in Figure 9 that the consoles of the e-Voting client and the e-Voting server graphical interfaces show only that the vote has been received. No information about any aspect at the client or at the server side is shown.

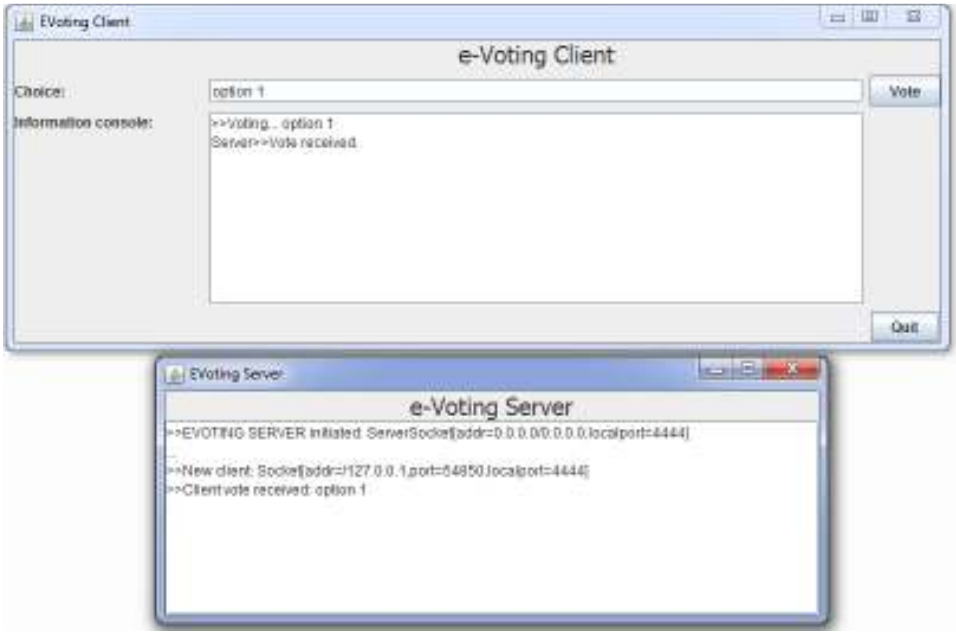

Figure 9. Information shown by the application wihouth aspects deployed.

4. **The client and the server deploy the security policies for the first scenario.**

After a negotiation, both the server and the client side of an application must deploy the new security policy. In this demonstrator we assume that the negotiation has already occurred, the Policy Interpreter module has both interpreted the new security policy and generated a SDS file. After this, the Aspect Generation module at both the client and the server side will be notified about that some adaptations need to be performed.

In order to initiate the adaptation at the server side (or at the client side) the user needs to:

a) *Select* the SDS file to be deployed for the first scenario in the Policy Interpreter Simulator (.\sdsFiles\escenario1Server.xml or .\sdsFiles\escenario1Client.xml) (see Figure 7).

b) Optionally, the user can *visualize* the file pressing the 'Visualize' button (see Figure 10 and Figure 11). On the one hand, Figure 10 shows the specification of the authentication security concept indicating its required functionality and parameters (the policy interpreter maps the security policies into security concepts and this is the

information that it is sent to the Aspect Generation module). This is the security concept to be deployed at the server side. On the other hand, Figure 11 shows the specification of the authentication, encryption and security timing security concepts indicating their required functionality and parameters and these are the security concepts to be deployed at the client side.

Figure 10. Example of the SDS file with the security concepts to be deployed at the server side

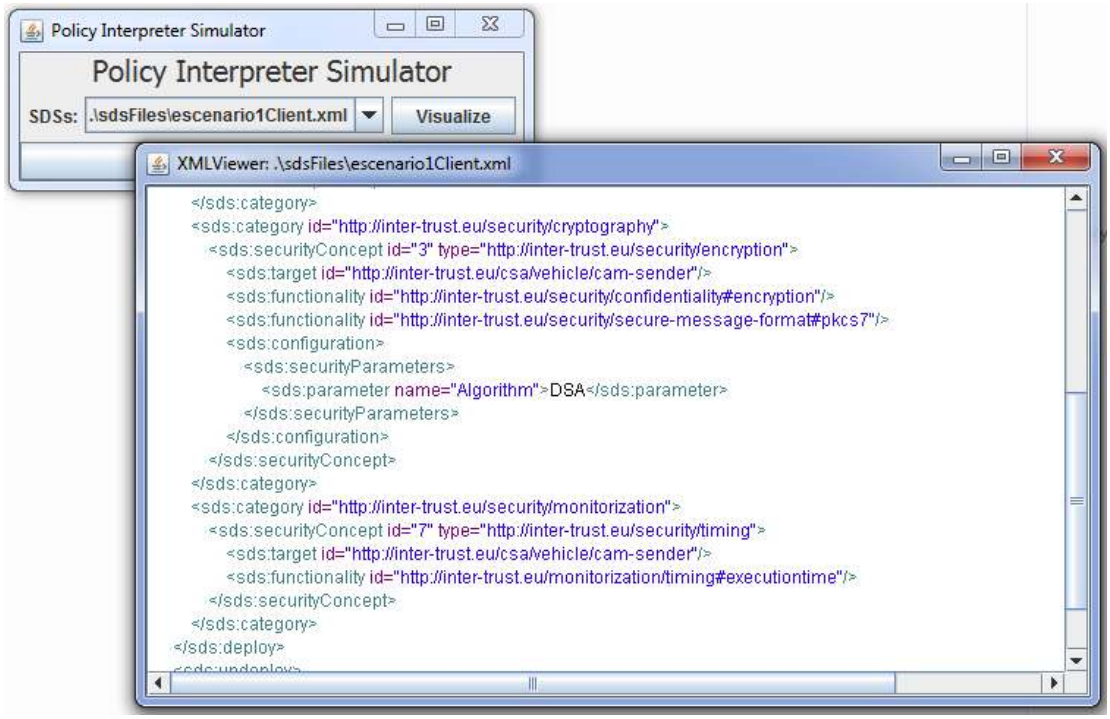

Figure 11. Example of the SDS file with the security concepts to be deployed at the client side

The format of these files is out of the scope of this document. The details can be found in deliverable D2.3.1.

c) *Notify* to the Aspect Generation module by pressing the 'Notify' button.

After doing this, the Policy Interpreter Simulator communicates with the Aspect Generation module and the demonstration of the AspectGeneration module begins to show how the notified security rules are automatically deployed in the application original code.

In order to facilitate the demonstration of the module, we have implemented the graphical interface shown in Figure 12.

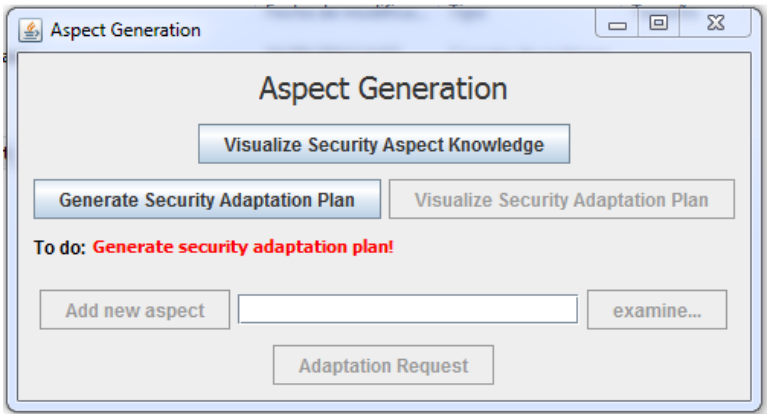

Figure 12. Grahical interface of the Aspect Generation module (for the purpose of the demonstrator)

Using this interface the user can:

a) Optionally, *visualize* the security aspectual knowledge (see the design of the software architecture of the Aspect Generation module in deliverable D4.2.1 for a detailed description of this information). For this first demonstrator we have implemented an initial version of the XML Schema that specifies the information that is required to define the mapping between the security policies (or the security concepts that appear in the SDS files previously shown) and the aspects (see Figure 13).

Figure 13. File with the mapping between security concepts and aspects

b) *Generate* the 'Security Adaptation Plan' by pressing the 'Generate Security Adaptation Plan' button (see Figure 12). This is the most interesting part of this module, which takes the last configuration being deployed and the new requested one and generates the difference among them. This difference is the security adaptation plan that consists on the list of aspects that need to be undeployed and/or deployed. This list of aspects is the output of the Aspect Generation module and the input to the Aspect Weaver module.

c) Optionally, *visualize* the security adaptation plan. For the deployment of the security policies of this first scenario, the security adaptation plan is shown in Figure 14 and Figure 15. We can observe that at the server side (Figure 14) the authentication aspect is deployed, while at the client side the authentication aspect, the encryption aspect and the aspect to monitor the time of encryption have to be deployed (Figure 15).

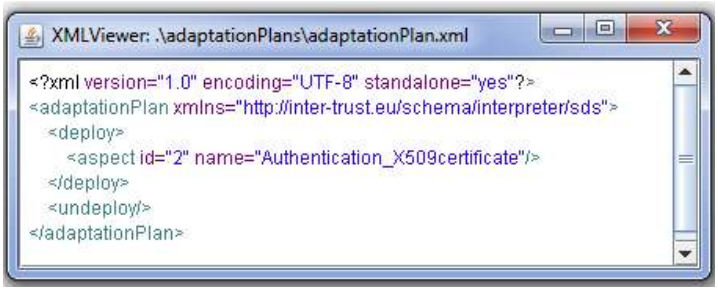

Figure 14. Security Adaptation Plan of the first scenario for the server

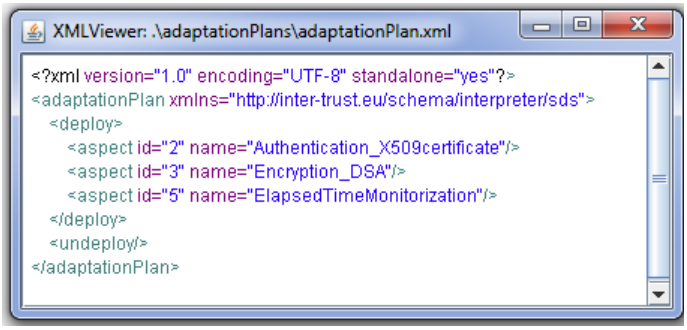

Figure 15. Security Adaptation Plan of the first scenario for the client

d) *Request* the adaptation by pressing the button 'Adaptation Request'. This makes the list of aspects previously generated to be sent to the Aspect Weaver module. We can say that the demonstrator of the Aspect Weaver begins here.

As part of the demonstrator of the Aspect Weaver module we have implemented a console that shows all the aspects that are deployed or undeployed by the aspect weaver (we are using for this first prototype the AspectJ weaver). The console for the server is shown in Figure 16, where we can see that the authentication aspect has been deployed, and the console for the client is shown in Figure 17, where we can see that the authentication aspect, the encryption aspect and the aspect to monitor the time required for encryption have been deployed.

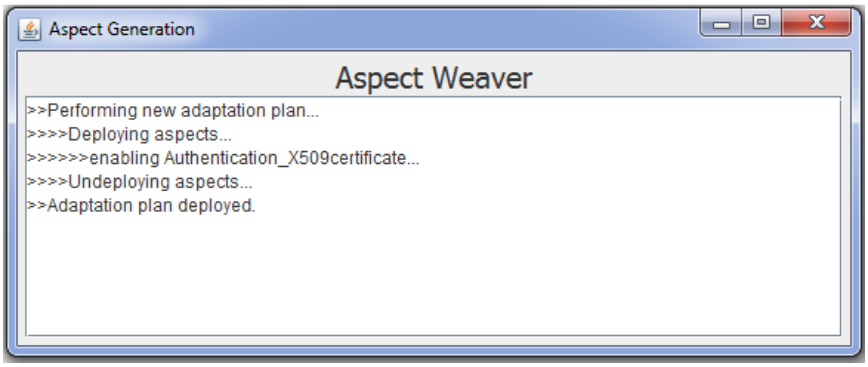

Figure 16. Aspect Weaver console at the server side

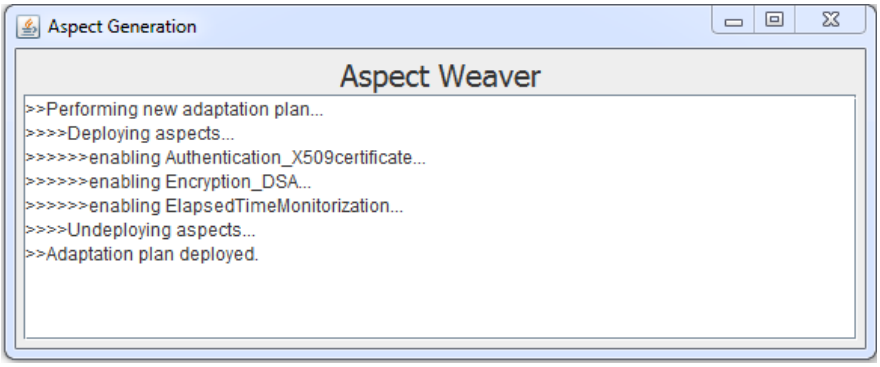

Figure 17. Aspect Weaver console at the client side

Now, when the client votes after the new security policy has been deployed we can observe that the information shown in the client and the server consoles is different, because now the aspects previously deployed are executed during the voting process (see Figure 18).

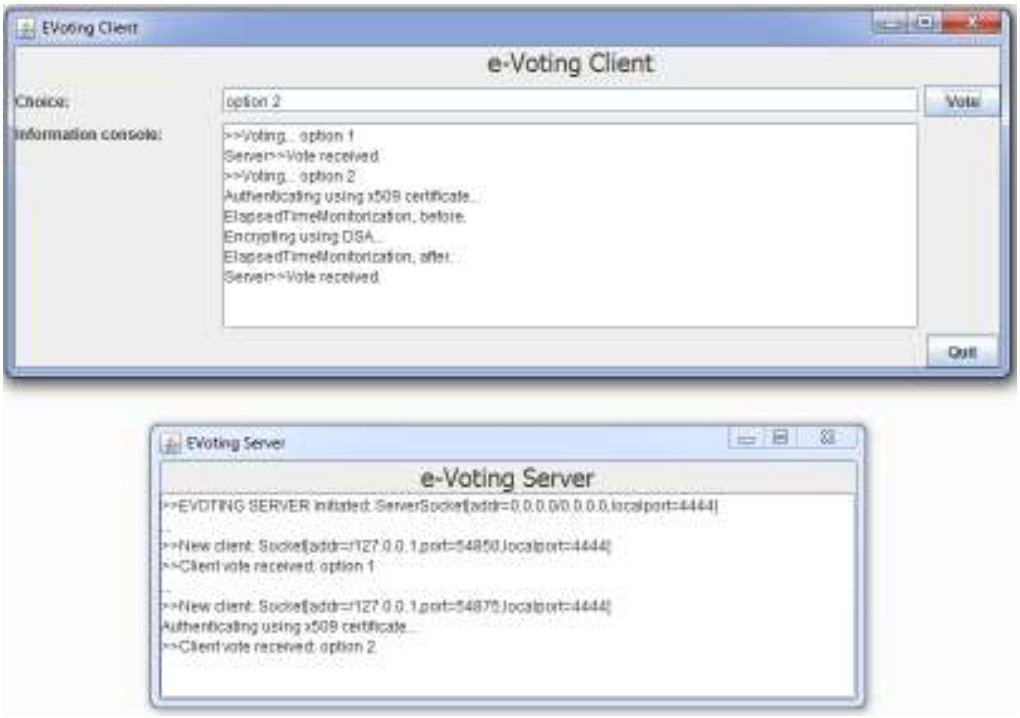

Figure 18. Consoles of the e-Voting client and the e-Voting server after the deployment of the security policy

5. **The same security policy is deployed in the client (or in the server).**

The interesting issue in this part of the demonstrator is to check that if a security policy is deployed that does not contain any modification regarding the previously deployed policy, the Aspect Generation module works correctly.

For instance, if we follow the same steps that in the previous deployment, and:

a) We deploy the security policy for the scenario 1 in the client again (by pressing the 'Notify' button in the Policy Interpreter Simulator, see Figure 7)

b) We generate the security adaptation plan using the graphical interface of the Aspect Generation module (see Figure 12).

c) We visualize the security adaptation plan (see Figure 12). The file must be empty as shown in Figure 19. The reason is that we are deploying a security policy that matches the security policy already deployed in the previous step and thus no changes are required (see Figure 19).

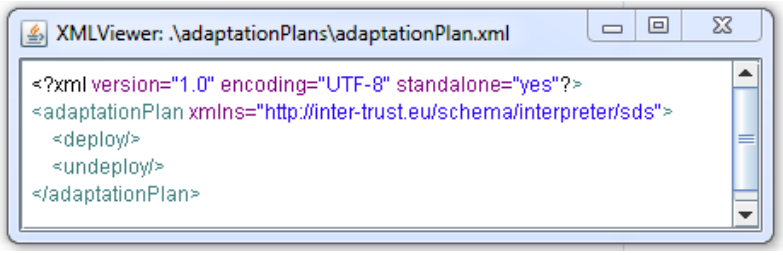

Figure 19. Empty security adaptation plan (no changes are required in the application)

d) If we execute the adaptation request no changes will be shown in the console of the Aspect Weaver module.

6. **The client and the server deploy the security policies for the second scenario.**

In order to deploy the security policies for the second scenario the user needs to follow exactly the same steps discussed in the deployment of the first scenario. The only difference is that in this case some existing aspects will be un-deployed while new ones will be deployed. Thus we only show here the security adaptation plans for both the client and the server side (see Figure 20 and Figure 21).

In this scenario we suppose that there are not enough resources at the client side to perform the encryption (this is notified by the ElapsedTimeMonitorization aspect that was deployed in scenario 1 with the purpose of measuring the time required by the client to cipher the vote). Thus, we assume that the client and the server have negotiated to delegate the vote encryption to the server. As a result of this negotiation and after receiving the notification of the Policy Interpreter module that a new adaptation of the application is requested, in Figure 20 we can observe that the Encryption aspect must be deployed at the server side. Similarly, in Figure 21 se observe that the encryption and the aspect to monitor the time required for encryption must be undeployed at the client side.

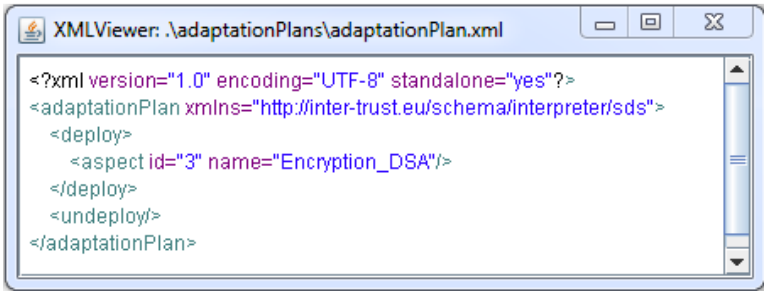

Figure 20. Security adaptation plan for the server in the second scenario

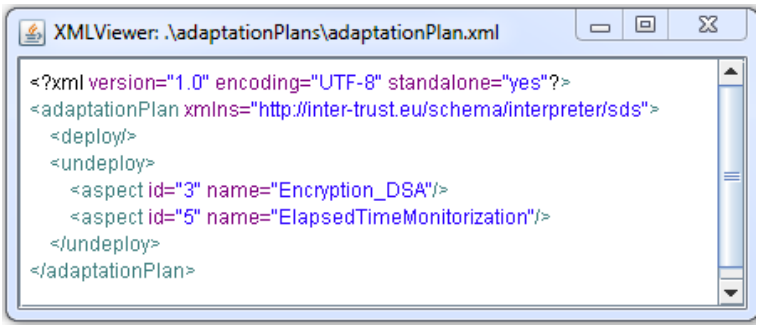

Figure 21. Security adaptation plan for the client in the second scenario

The execution of these security adaptation plans by the Aspect Weaver module is shown in Figure 22 (we see how the encryption aspect have been deployed) and Figure 23 (we see how the encryption and the monitoring aspect have been undeployed).

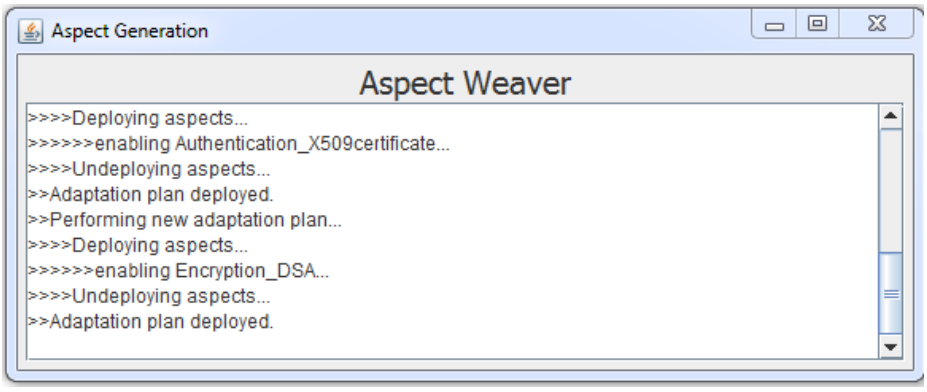

Figure 22. Aspect Weaver console for the server in the second scenario

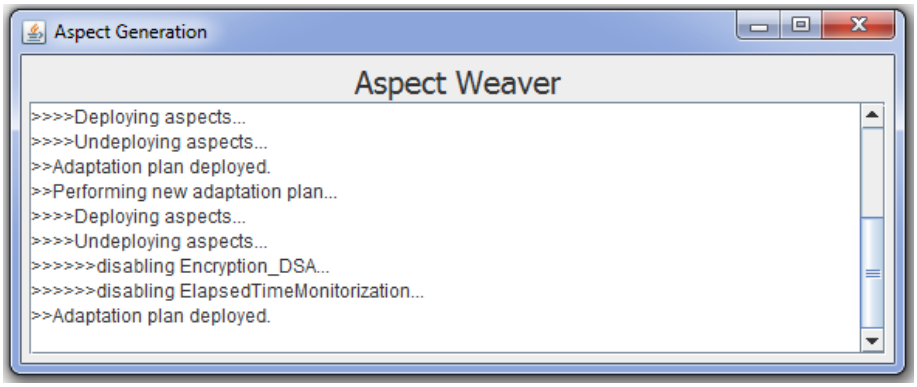

Figure 23. Aspect Weaver console for the client in the second scenario