Chapter 6

Implementation Results and Performance Analysis

This chapter discusses the important security related scenarios that have been implemented through the security policy framework and also presents performance analysis of the individual authentication, privacy, trust and authorization policies. The implementation has been done in .NET environment with the support of WSE 3.0 toolkit. The implemented security services have been exposed as web services. Today the world is witnessing the convergence of grid and web services. The two (grid and web) started far apart in specifications, technology and applications but now they are converging into a common set of standards and specifications. So security services have been implemented as web services. WSE 3.0 implements many web services security specifications like WS-Security, WS-SecureConversation, WS-Trust *etc.* We have made use of these specifications to implement the framework. For other specifications like XACML and SAML, we are not using any software instead the necessary functionality has been developed and used wherever required. All security information is exchanged as SOAP messages. SOAP messages are constructed and WS-Security information is embedded using WSE 3.0 toolkit. The other web services security specifications like WS-Trust and WS-SecureConversation have been used for security token exchange and to establish secure communication contexts. Using web services security specifications, we are also guarantying the confidentiality and integrity of the messages exchanged as WSE 3.0 provides features for encrypting selected parts of SOAP messages as well as digital signatures. All types of security policies have been expressed in simple XML or XACML and are stored in the policy database. The XML database has been developed in MS-SQL Server. To implement specific aspects of the security policy framework, we are making extended use of XACML and other specifications.

For implementation purpose, the .NET based approach has been chosen to increase the participation of .NET programmers and communities in the development and use of grid applications. Though most of the grid applications and infrastructures in use today are based on Unix/Linux platform but initiatives in the direction of .NET based approaches to grid application development are emerging. *e.g.* Alchemi [93] is a desktop grid framework based on .NET. The caBIG (cancer Biomedical Informatics Grid) [157] deals with creating a design to support caBIG and caGrid [158] (underlying platform that provides the basis for connectivity of caBIG tools) using .NET. NGrid [159] is an open source grid computing framework written in c#. GridFTP [160] is a data transfer protocol for accessing distributed data on the Grid using .NET. WSRF.NET [161] is an implementation of full set of specifications for WSRF and WS-Notification on the .NET framework. OGSI.NET [162] is the implementation of Open Grid Services Infrastructure (OGSI) on .NET framework.

The .NET based implementation of grid security can complement the .NET approaches and projects described above. .NET based approach also supports rapid application development, involves less installation, configuration and development effort, is good GUI based and provide easy error handling and debugging facilities. The proposed models are not platform/hardware dependent. The proposed models can be implemented on UNIX based systems as well. Following sections detail about scenarios implementation and performance analysis.

6.1 Scenarios Implementation

This section presents the important security related scenarios that have been implemented through the security policy framework. The support for these scenarios is generally required in most of the grid and web based systems. Microsoft and IBM in their joint paper "Security in Web Services World: A Proposed Architecture and Roadmap" [33] have described many important security related scenarios that must be addressed by a security implementation. The security policy framework supports all these scenarios. We have divided the implemented scenarios into four categories namely authentication, privacy, trust and authorization related scenarios. Following paragraphs discuss each of these categories.

6.1.1 Scenarios related to Authentication

The security policy framework is capable of implementing single sign-on and delegation along with non-repudiation, confidentiality, integrity of exchanged information and secure conversation. Single sign-on and delegation features are supported through proxy certificates. The other security requirements like non-repudiation, confidentiality, integrity of exchanged information and secure conversations are supported through XML-Signature, XML-Encryption, WS-Security and WS-SecureConversation specifications. WSE 3.0 toolkit under .NET environment has been used to incorporate the functionality of WS-* specifications in implemented scenarios. Figure 6.1 and Figure 6.2 represent single sign-on and delegation scenarios.

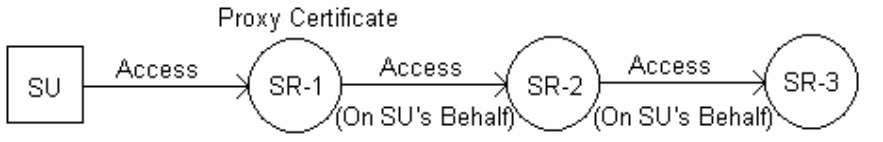

Figure 6.1: Schematic showing Single Sign-On

In Figure 6.1, Subject SU issues proxy certificate to Service SR-1 so that SR-1 can act on its behalf. SR-1 uses this proxy certificate to get authentication at SR-2 on SU's behalf. Similarly, using the same concept, Subject SU also gets authenticated at SR-3 by SR-2. To generate proxy certificates, the sequence of steps described in Table 5.1 have been used.

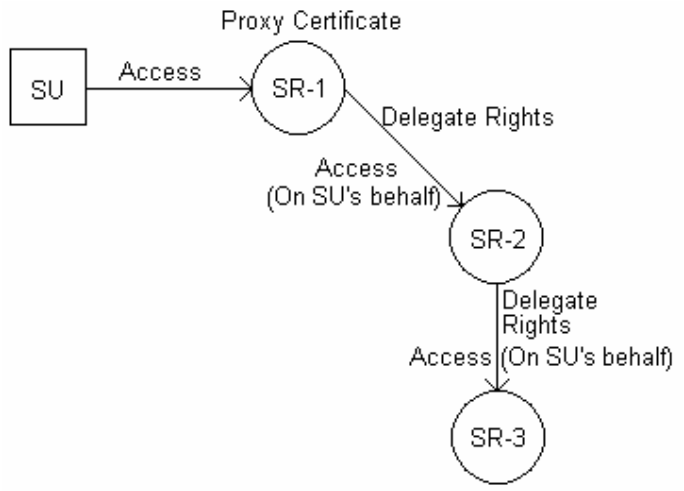

Figure 6.2: Schematic showing Delegation and Single Sign-On

Figure 6.2 represents delegation scenario. Here Subject SU passes some/all of his access rights to SR-1 using proxy certificates. The delegated rights are embedded in proxy certificates through PCI (Policy Certificate Information) field. In this case SR-1 can access SR-2 on SU's behalf but only the actions implied by delegated rights can be performed. The delegated rights can further be delegated if the original subject has passed the right of further delegation in the delegation list.

6.1.2 Scenarios related to Privacy

The security policy framework is capable of implementing anonymous access and privacy based access to services/resources. It also supports the access of hidden services through custom security credentials. Figure 6.3, Figure 6.4 and Figure 6.5 represent scenarios related to access of private information, anonymous access and hidden services access, which have been implemented through the framework.

In Figure 6.3, Subject SU sends his private information to Service SR-1 and both establish and agree on a privacy relationship. If SR-1 exposes SU's private information then access to this information is provided only if the requester (in this case SR-2) conforms to established privacy policies and relationships. To implement this scenario, the privacy based access mechanism as presented in Table 5.4 has been used.

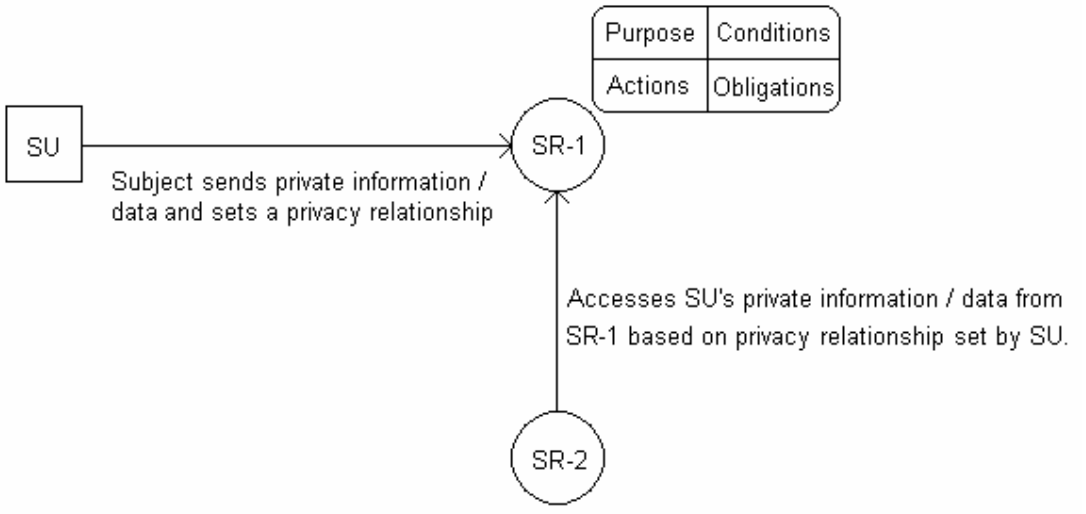

Figure 6.3: Schematic showing access of private information based on established privacy policies and relationships

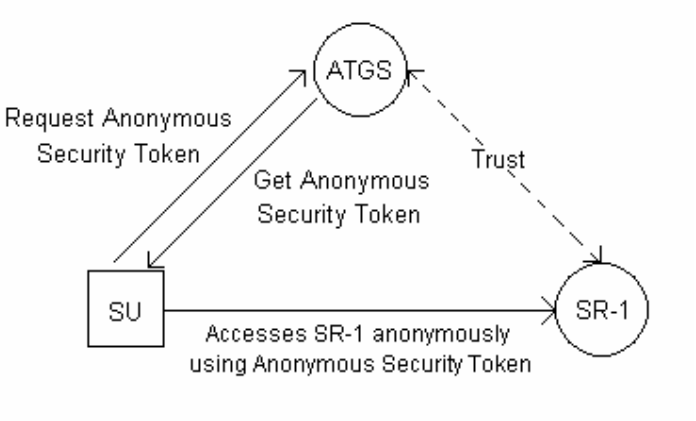

Figure 6.4: Schematic showing anonymous access of a service

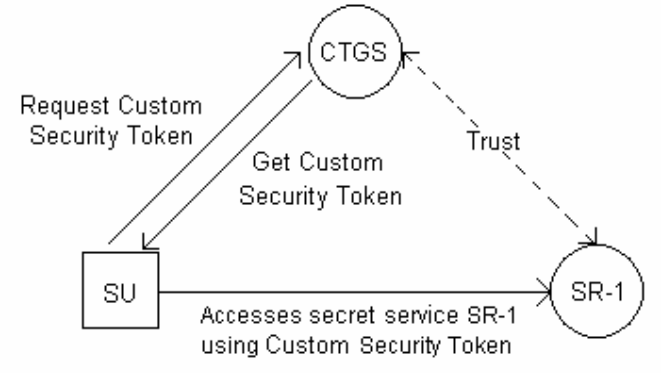

Figure 6.5: Schematic showing access of a hidden service

Figure 6.4 shows anonymous access of a service. In this case, Subject SU requests anonymous security token from an ATGS (Anonymous Security Token Granting Service) and presents it to service SR-1 in order to access it anonymously. Here a prior trust relationship exists between ATGS and SR-1. The anonymous security token asserts about authorization information of the subject but removes its identity. Thus subject can access the service anonymously.

Figure 6.5 shows how access to a secret service is provided. In this case SU requests custom security token from Custom Security Token Granting Service (CTGS) and presents it to SR-1 to access it. As custom security token is generated by CTGS, SU has no idea of the information present in custom security token. SU does not know which of his attributes / other information is embedded in this token. SU simply presents the custom security token to SR-1 and access the service but does not know exactly which of his attributes/credentials are causing access to this service. Thus SU can access the

service without knowing the exact security requirements of the service. The anonymous security tokens and custom security tokens have been generated through tools available in .NET environment.

6.1.3 Scenarios related to Trust

The security policy framework is capable of determining the trustworthiness of target service/resource through direct as well as recommended trust. Figure 6.6 and Figure 6.7 show scenarios related to direct and recommended trust which have been implemented.

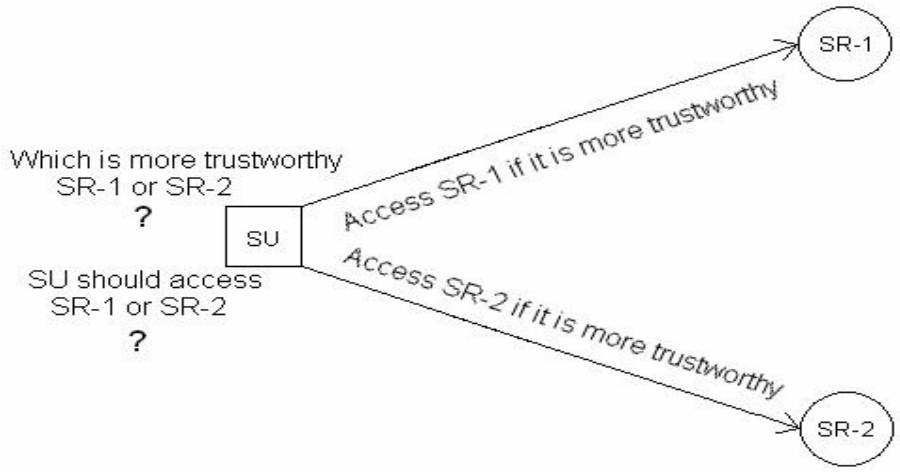

Figure 6.6: Schematic showing determining direct trust with a service

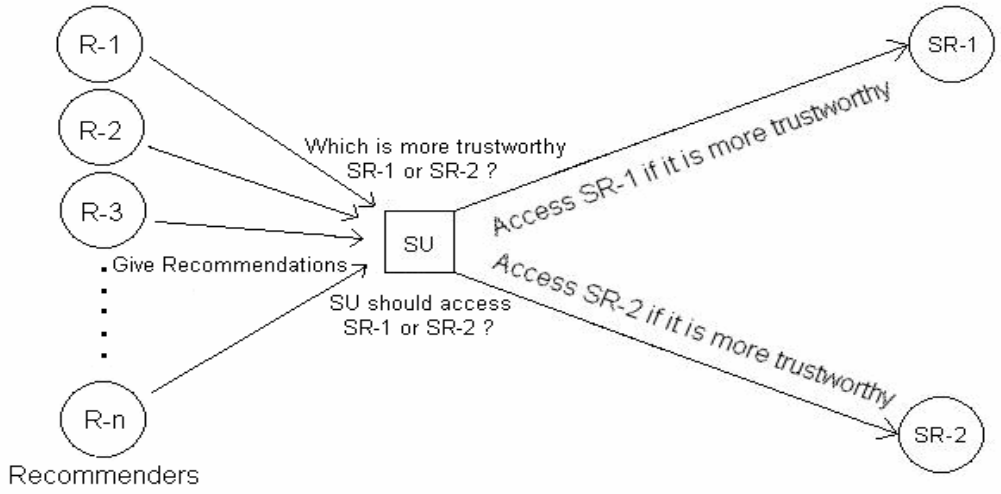

Figure 6.7: Schematic showing determining direct and recommended trust with a service

In Figure 6.6, Subject SU determines the trustworthiness of different services (in this case SR-1 and SR-2) exposed in the environment and based on the result accesses a particular service which comes out to be more trustworthy. In determining trust on a service, subject can also take help from his recommenders. This is shown in Figure 6.7. After determining the direct and recommended trust, the subject makes a final decision regarding which service to access. To calculate direct and recommended trust, the pseudo codes presented in Table 5.8 and Table 5.9 have been used. The framework implements these trust related scenarios. Different variations of these scenarios can also be implemented through the framework using other services (listed in Table 5.10), exposed by the trust model.

6.1.4 Scenarios related to Authorization

The security policy framework is capable of implementing distributed access and policy based access. Figure 6.8, Figure 6.9 and Figure 6.10 represent scenarios related to policy based access, distributed authorization and federation of security services that have been implemented through the framework.

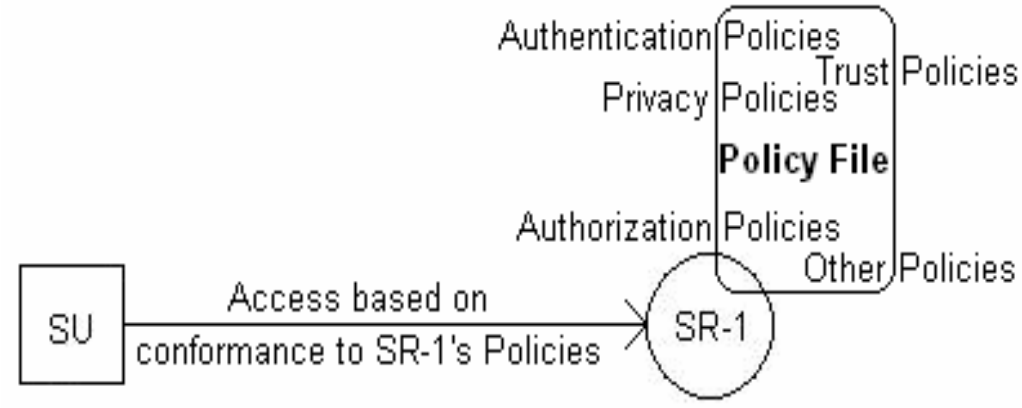

Figure 6.8: Schematic showing access based on conformance to service's policies

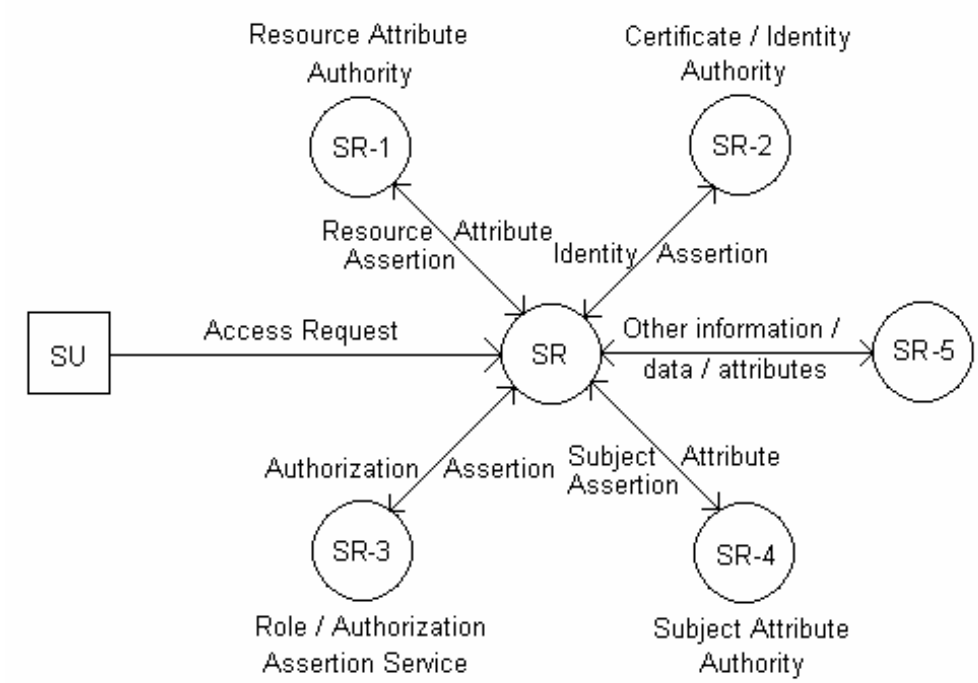

Figure 6.9: Schematic showing scenario related to distributed authorization

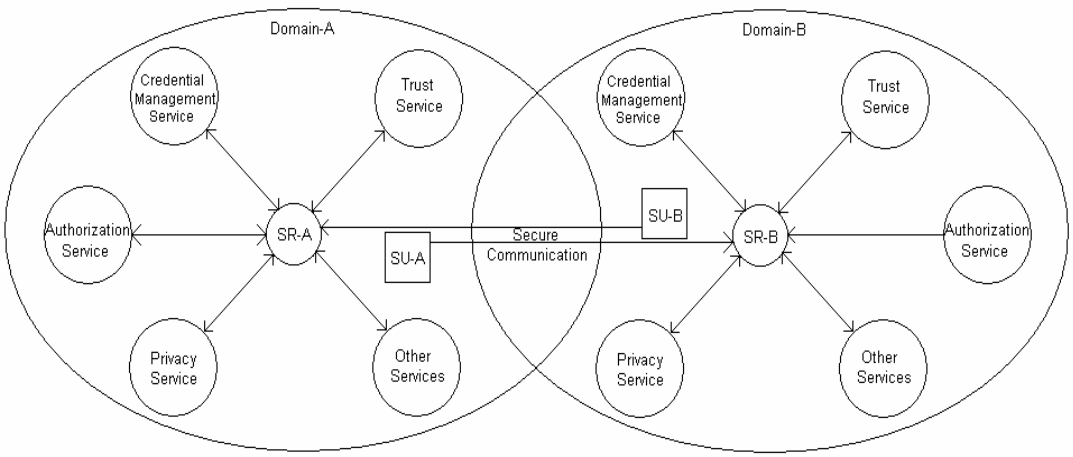

Figure 6.10: Schematic showing scenario related to federation of security services

In Figure 6.8, Subject SU generates access request to access service SR-1. Access to this service is provided based on conformance to its policies. To implement this scenario, the sequence of steps presented in Table 5.11 have been used. Figure 6.9 represents distributed authorization. To determine authorization information and attributes, a service

takes the help of other services. The service receives identity assertions, subject and resource attribute assertions and other information/data from different authorities distributed in the environment. These assertions are used to prepare final authorization result. Figure 6.10 shows the federation of security services. Subject of one domain access the services of other domains which are protected by security services of their own domain. All these scenarios have been implemented successfully. The security policy framework is also capable of implementing variations of these scenarios.

6.2 Performance Analysis

This section presents the performance analysis of individual authentication, privacy, trust and authorization policies. The implementation environment consists of 50 domains with users ranging from 10 to 50 in each domain. All the domains have more than 10 service providers that provide different services/resources to other domains. Resources have been exposed as services. Subjects have been given different security credentials (X.509 certificate, username: password, Kerberos *etc.*). These credentials have been generated through .NET tools. Database contains authentication, privacy, trust and other security policies established among subjects and services of different domains. Policy database is maintained by every domain for its subjects. Each service is associated with a policy file that describes its service policy. The security policies are exposed by services in the environment. Access to services/resources is provided after conformance to these policies. For the performance analysis of different policies, we have created a sample service and attached to it the policies related to the type (authentication, privacy, trust and authorization) which is being analyzed. For each different type, we have noted the time taken by the framework in evaluating individual policy of that type and then the set of policies of that type by varying the number of policies in the policy file.

Time taken by different components *i.e.* PIP, PDP and PEP in evaluating each type of policy has been noted separately. Time taken by PIP component represent the time it takes to fetch, interpret and supply subject, resource, environment and other attributes to PDP for the evaluation of a policy. Time taken by PDP component represent the time taken to fetch, interpret and evaluate applicable policy. Note that PDP time includes PIP time also. Time taken by PEP component represent the time to prepare authorization decision query, passing it to PDP and receiving the response. Thus it includes PDP time

also and represents the overall time taken to enforce a policy. As PEP's task is to enforce policy according to response received from PDP and PDP being the major component for PEP in evaluating policy, there is not much difference in PDP time and PEP time. Another thing to note is that the function of privacy handler and trust handler is also like PDP component as in the framework, they are being used to evaluate privacy and trust policies specifically. This is clear from Figure 5.9 also where privacy and trust handler components have been shown as privacy and trust services parallel to different PDP services. In the following paragraphs, for all types of policies, we have used the term PDP to represent the component responsible for evaluating policies.

Different implementations have been tested on systems of different configurations, with memory ranging from 256MB to 1GB and processor speed ranging from 900MHz to 2GHz, and same result patterns have been observed. Moreover the execution has also been performed by varying the number of entities (domains, service providers, services, subjects *etc.*). In this case also the same result patterns have been observed. Graphs of following sections shows the result of implementation on most constrained system *i.e.* system having 256MB of RAM and 900MHz of CPU. The number of entities (domains, service providers, services, subjects *etc.*) used in the implementation are same as mentioned in the beginning of this section. The comparison of different types of policies with each other has also been done with respect to time taken by them for their enforcement. Following sections present these details.

6.2.1 Performance analysis of authentication policies

For the performance analysis of authentication policies, 50 pure authentication related policies have been created and attached to the service. These policies cover different aspects of authentication like the security credentials required by the service, encryption and signature requirements, single sign-on and delegation requirements of the service *etc.* Then the time taken by each of these policies to get evaluated individually is noted. The time taken by PIP, PDP and PEP components is also noted separately. The graphs of Figure 6.11, Figure 6.12 and Figure 6.13 show these details. For authentication policies, evaluation functionality is provided by the authentication handler which has been implemented in PDP itself. The average time taken by PEP component for authentication policies comes out to be 53.38 ms. There is not much difference in time taken by PEP

component to evaluate different authentication policies. This is because different types of authentication policies involve similar type of processing and almost all the information required to evaluate authentication policy is available from the access request itself. The resource, environment and other attribute access requirements in evaluating authentication policies are minimal. This is clear from the PIP time also as shown in Figure 6.11, which comes out to be 0.32 ms.

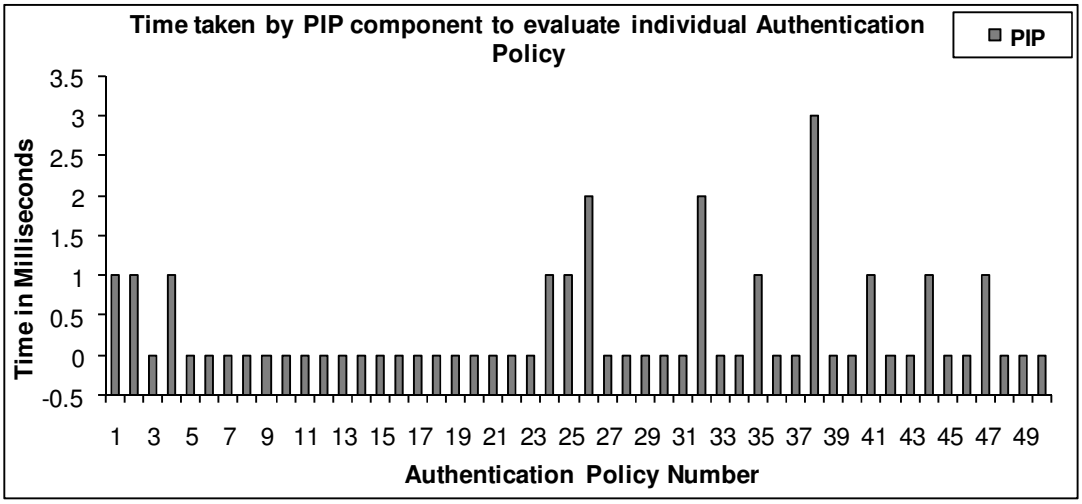

Figure 6.11: PIP time to evaluate individual authentication policy

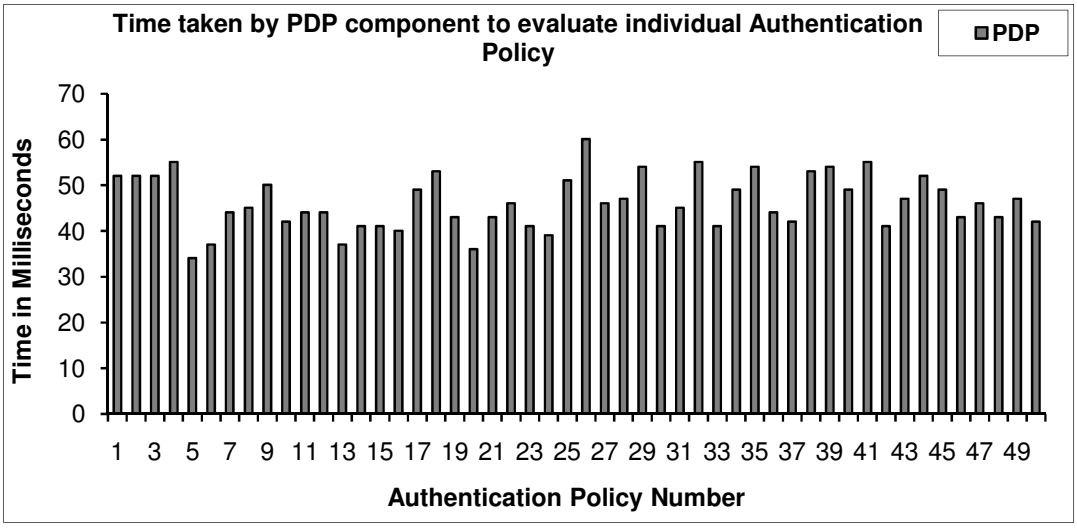

Figure 6.12: PDP time to evaluate individual authentication policy

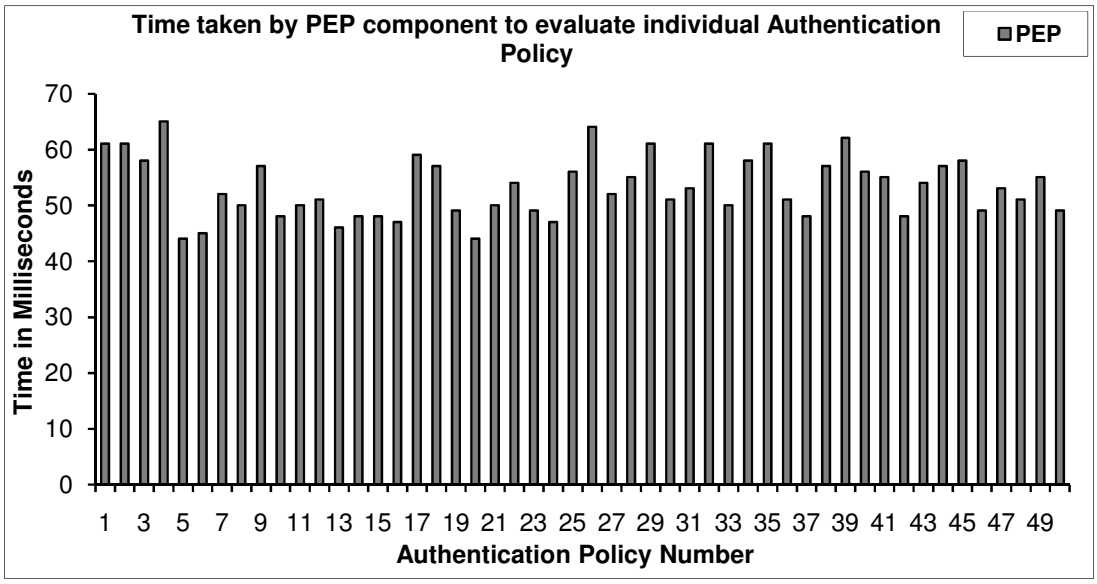

Figure 6.13: PEP time to evaluate individual authentication policy

In the second phase, the number of authentication policies attached with the service have gradually been increased and the time taken by PIP, PDP and PEP components is noted. Graphs of Figure 6.14, Figure 6.15 and Figure 6.16 show these details.

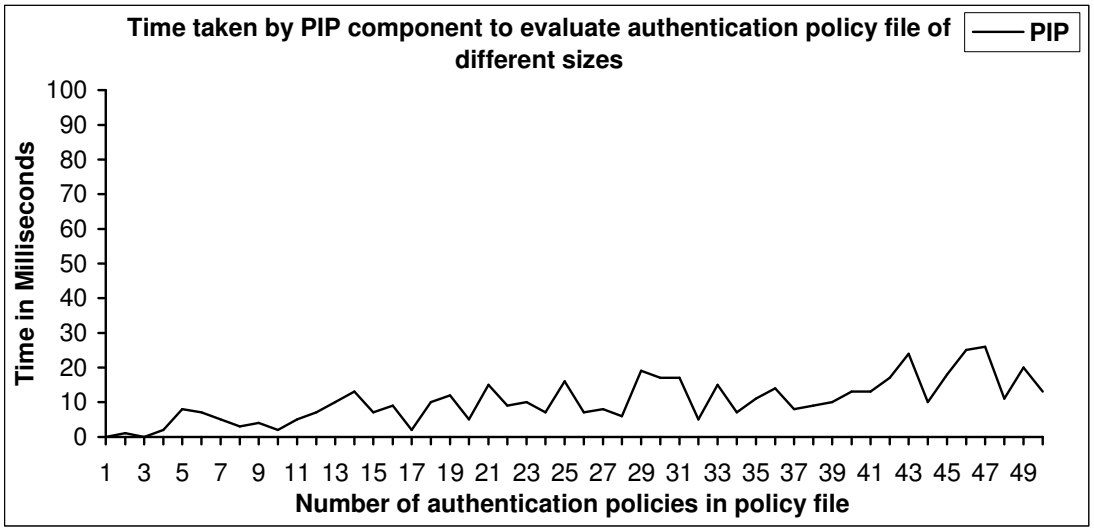

Figure 6.14: PIP time to evaluate authentication policy file of different sizes

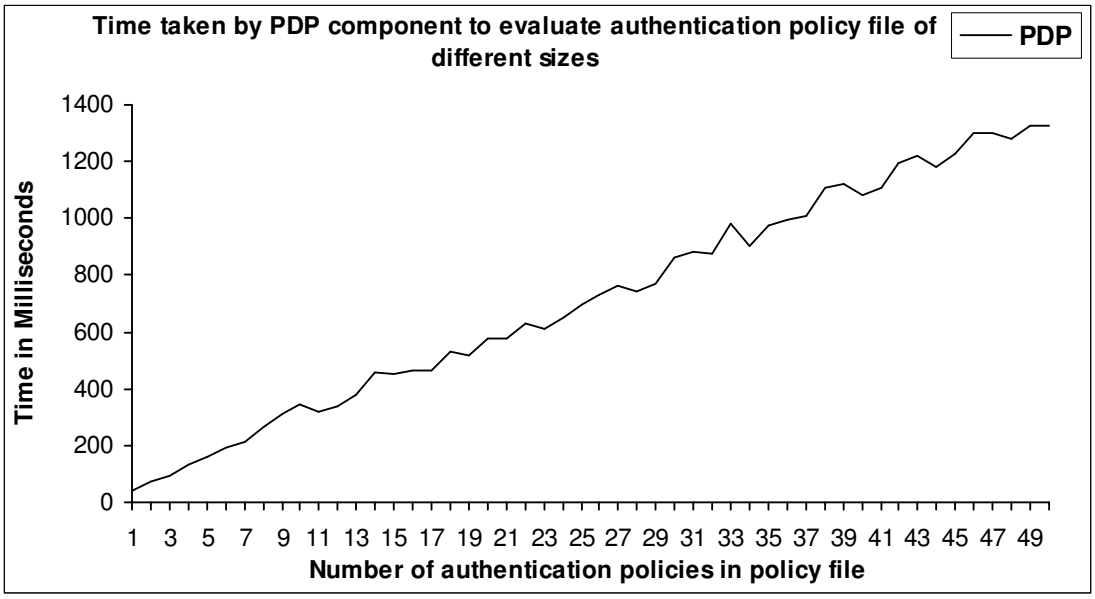

Figure 6.15: PDP time to evaluate authentication policy file of different sizes

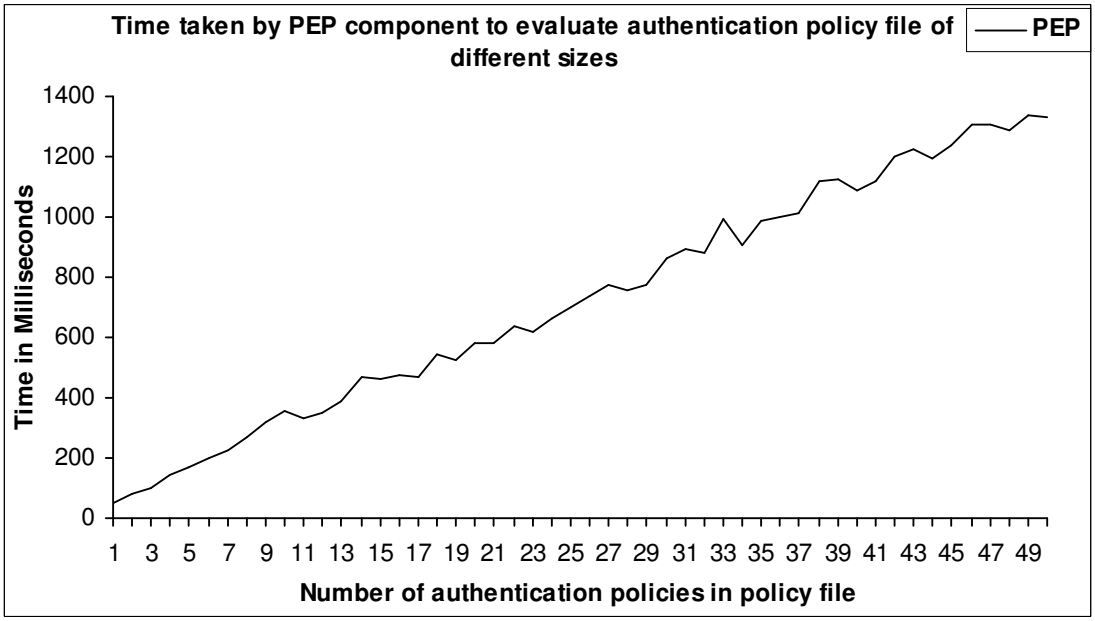

Figure 6.16: PEP time to evaluate authentication policy file of different sizes

The attribute access requirements in this case also remained minimal and did not increase significantly by increasing the number of authentication policies. On the other hand, the time taken by PEP and PDP components increased linearly with the increase in number of authentication policies. This is what we were also expecting. It is clear from

the graphs of Figure 6.14, Figure 6.15 and Figure 6.16 that increasing the number of authentication policies attached with a service does not adversely affect the performance of the authentication framework.

6.2.2 Performance analysis of privacy policies

For the performance analysis of privacy policies, 50 privacy related policies have been identified that cover different aspects related to privacy like purpose based access, hidden service access, anonymous service access, access of private information *etc.* Then the time taken by PIP, PDP and PEP components to evaluate individual privacy policy is noted. For privacy policies, PDP functionality is provided by privacy handler. The average time taken by PIP, PDP and PEP components for privacy policies comes out to be 24.65 ms, 73.58 ms and 81.34 ms respectively. Compared to authentication policies, more access to subject, resource and environment attributes is noted. This is because privacy based access involves processing of private information and policies that make use of these attributes. The details of time taken by different components are shown in the graphs of Figure 6.17, Figure 6.18 and Figure 6.19.

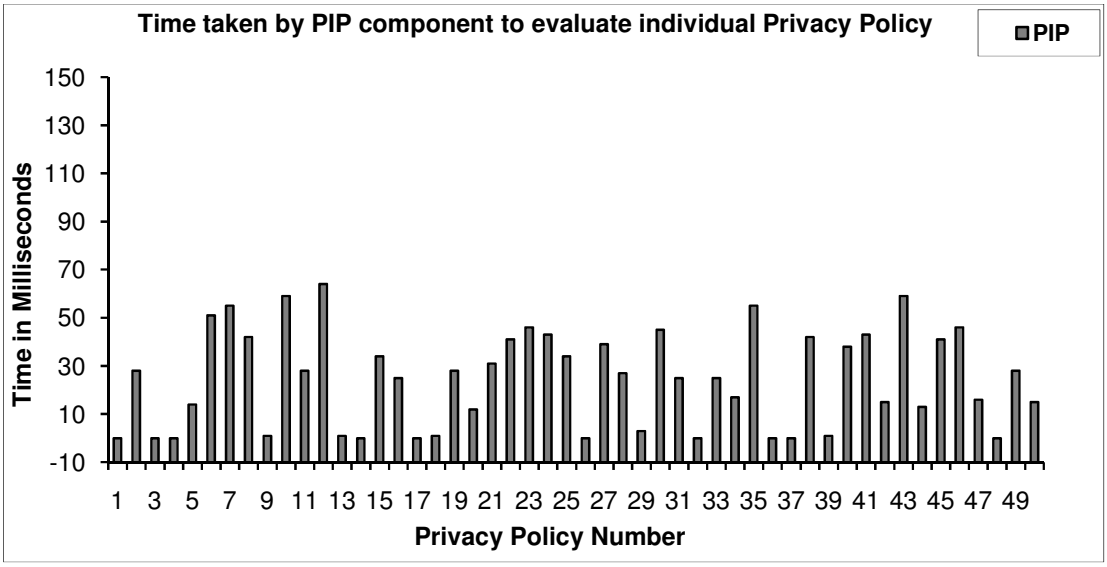

Figure 6.17: PIP time to evaluate individual privacy policy

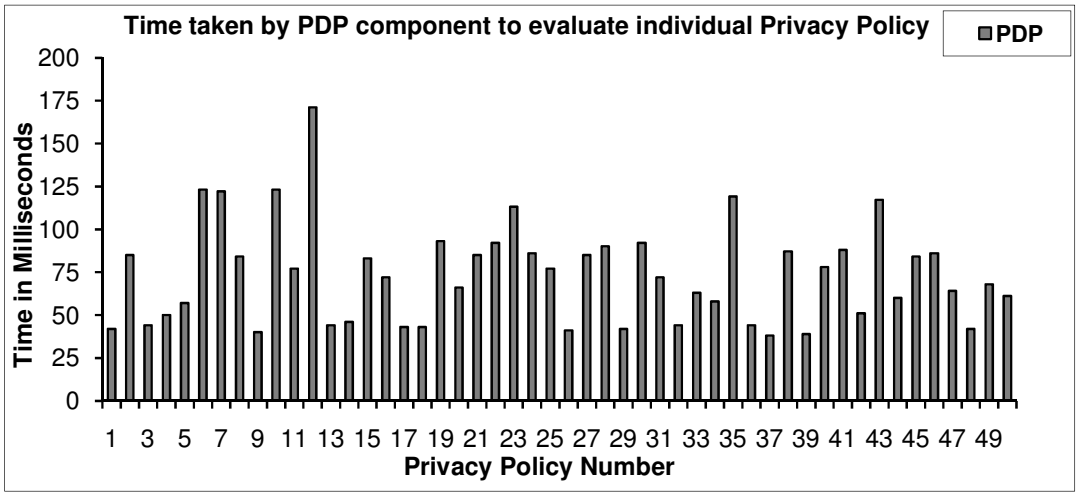

Figure 6.18: PDP time to evaluate individual privacy policy

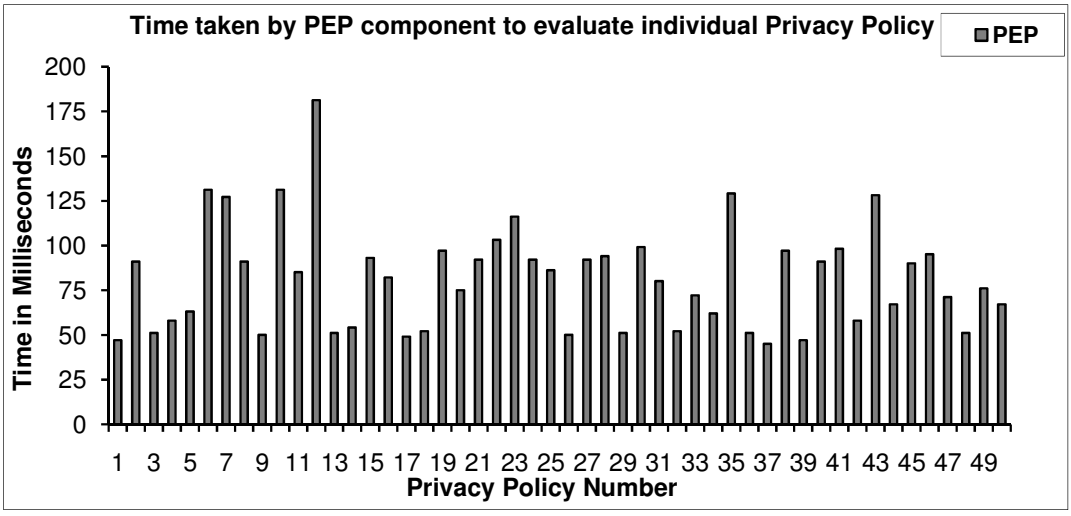

Figure 6.19: PEP time to evaluate individual privacy policy

Next the time taken by PIP, PDP and PEP components to evaluate different number of privacy policies is noted. The time taken by all the components increases linearly and not exponentially with the increase in number of privacy policies attached with the service/resource. This shows that privacy model also does not adversely affect the performance of authorization framework with the increase in number of privacy policies. The details are shown in the graphs of Figure 6.20, Figure 6.21 and Figure 6.22.

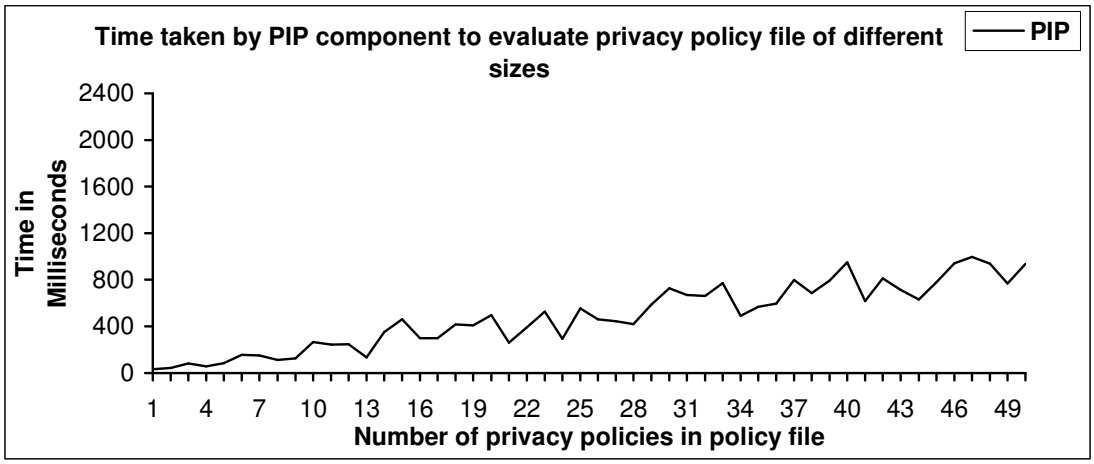

Figure 6.20: PIP time to evaluate privacy policy file of different sizes

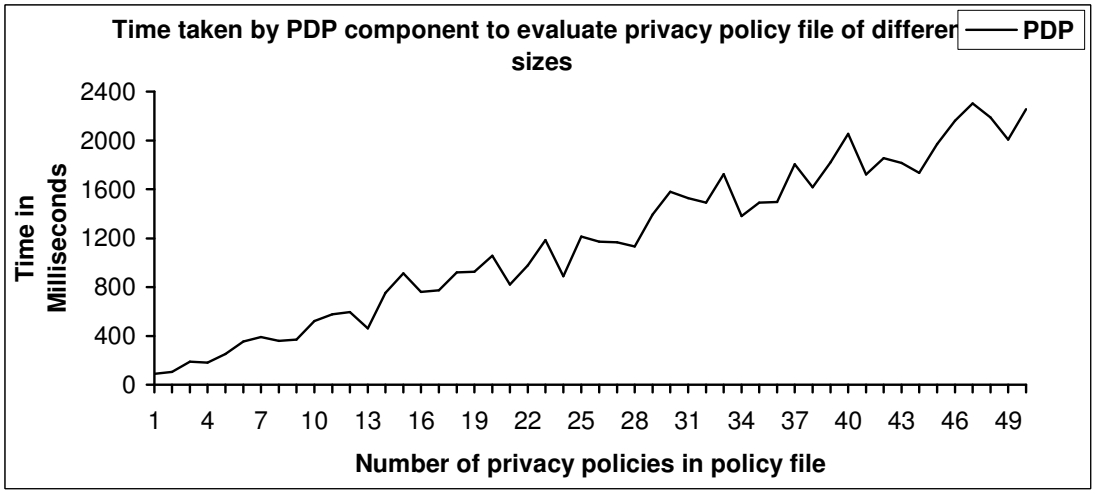

Figure 6.21: PDP time to evaluate privacy policy file of different sizes

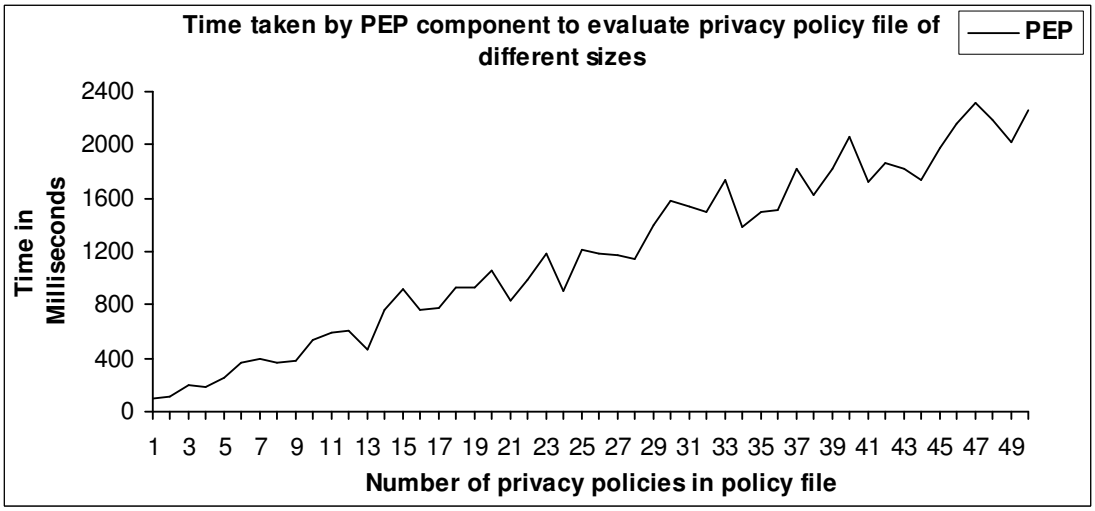

Figure 6.22: PEP time to evaluate privacy policy file of different sizes

6.2.3 Performance analysis of trust policies

Like authentication and privacy policies, for the performance analysis of trust policies, 50 trust related policies have been created. These policies include different aspects related to trust and involve determining trustworthiness of domain, service provider or service. Then these policies have been attached to the service and the time taken by PIP, PDP and PEP components for each trust policy is noted. The average time taken by PEP component in this case comes out to be 178.16 ms which is higher than the average time taken by PEP component for individual authentication and privacy policy. This is because the determination of trust value of target (service, service provider or domain) involves calculations and access to history of past interactions taken place between source and target. The average time taken by PIP component is also higher compared to time taken by PIP component for authentication and privacy policies as trust policies make more use of subject, resource, environment and other attributes. Graphs of Figure 6.23, Figure 6.24 and Figure 6.25 show these details.

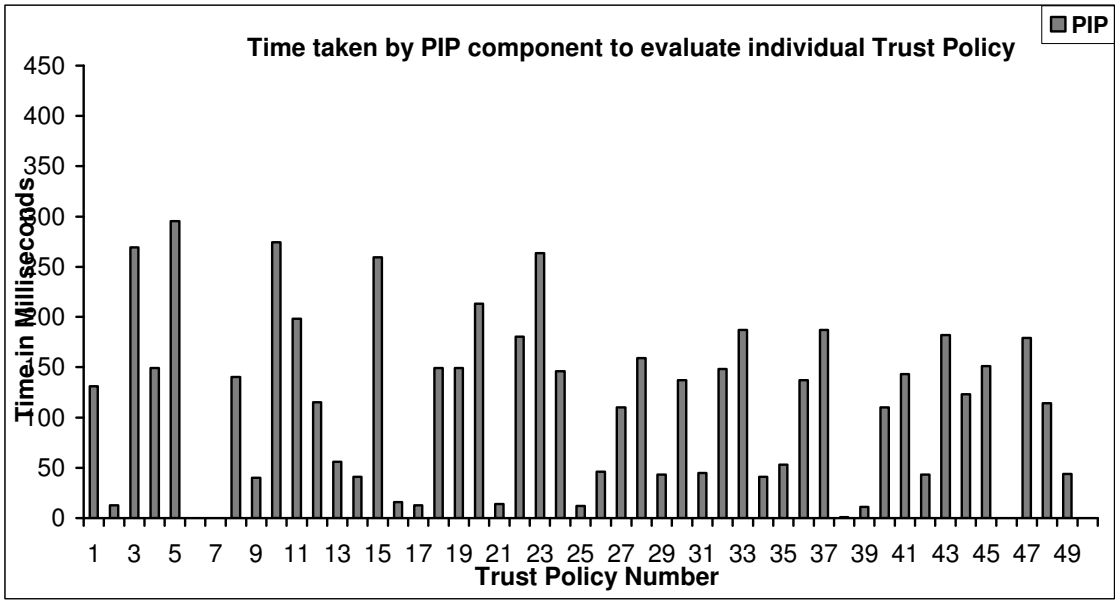

Figure 6.23: PIP time to evaluate individual trust policy

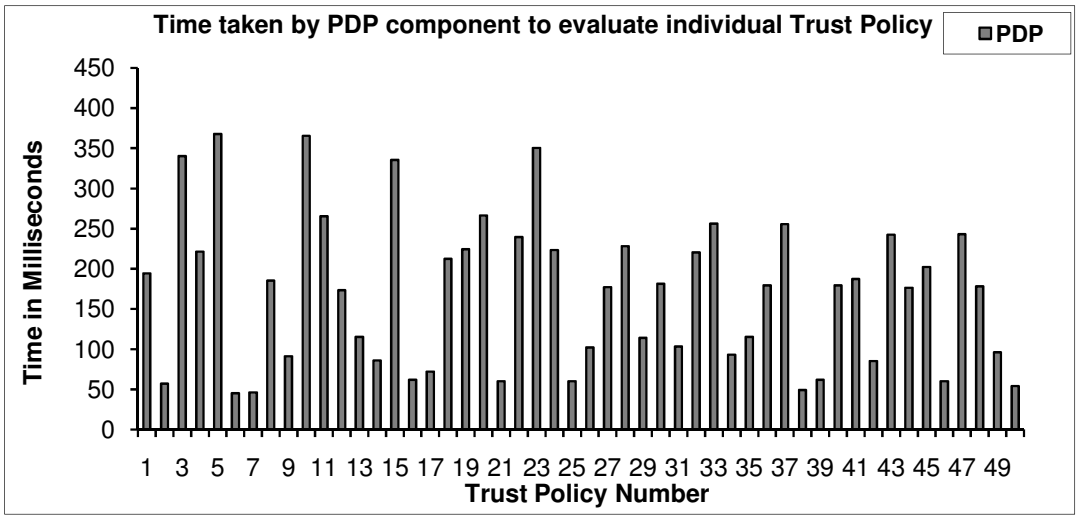

Figure 6.24: PDP time to evaluate individual trust policy

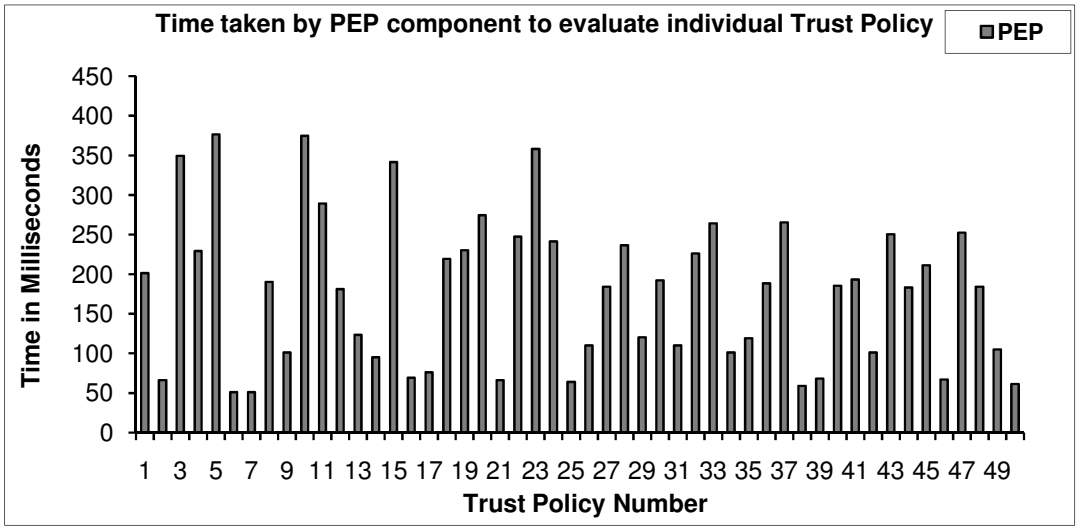

Figure 6.25: PEP time to evaluate individual trust policy

Next the time taken to evaluate different number of trust policies is noted. For this the number of trust policies attached with the service have been varied form 1 to 50 and the time taken by PIP, PDP and PEP components have been noted. The same behavior pattern as appeared with authentication and privacy policies is observed but in this case the PIP, PDP and PEP time increases more assertively with the increase in number of trust policies. This is again due to the fact that determination of trust value of a target involves calculations and access to history of past interactions taken place between source and

target, which is time consuming. Therefore, performance can be a concern if the number of trust policies attached with a service/resource are more. Graphs of Figure 6.26, Figure 6.27and Figure 6.28 show these details.

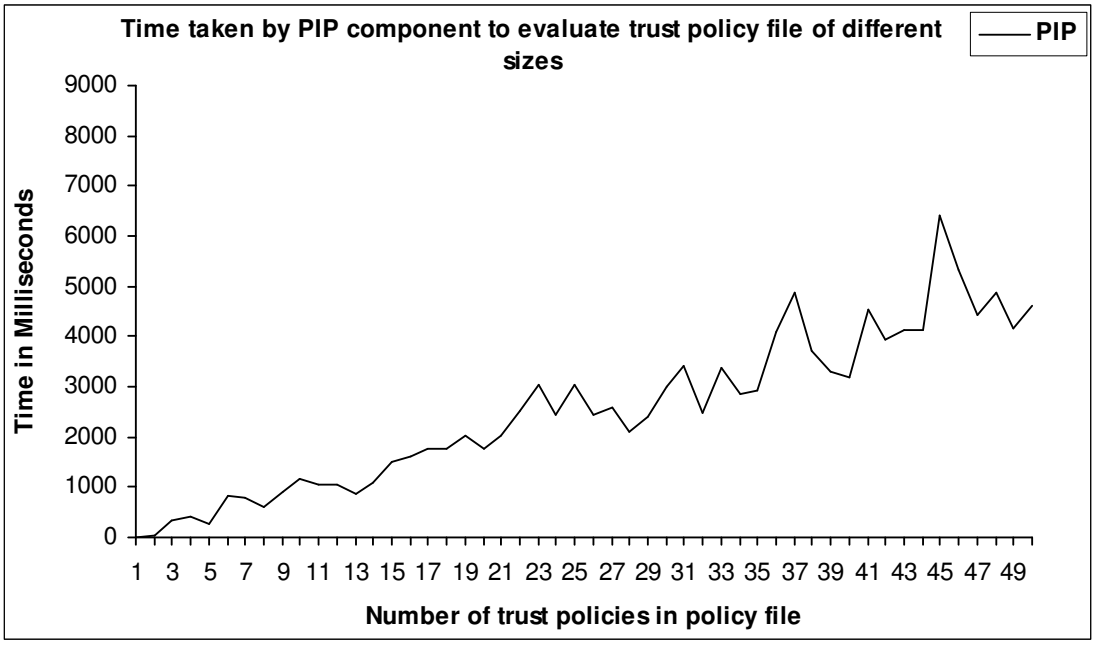

Figure 6.26: PIP time to evaluate trust policy file of different sizes

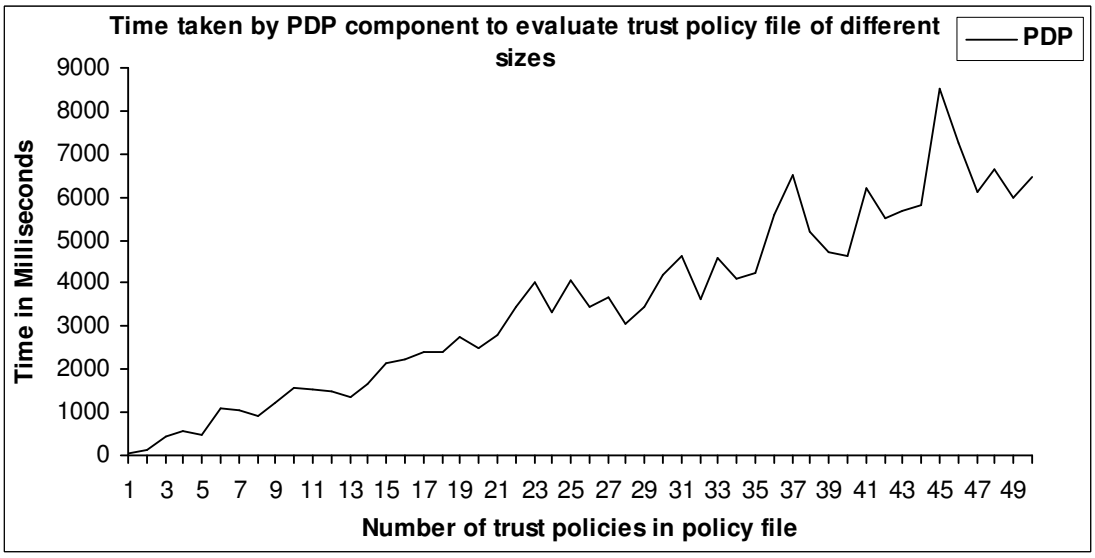

Figure 6.27: PDP time to evaluate trust policy file of different sizes

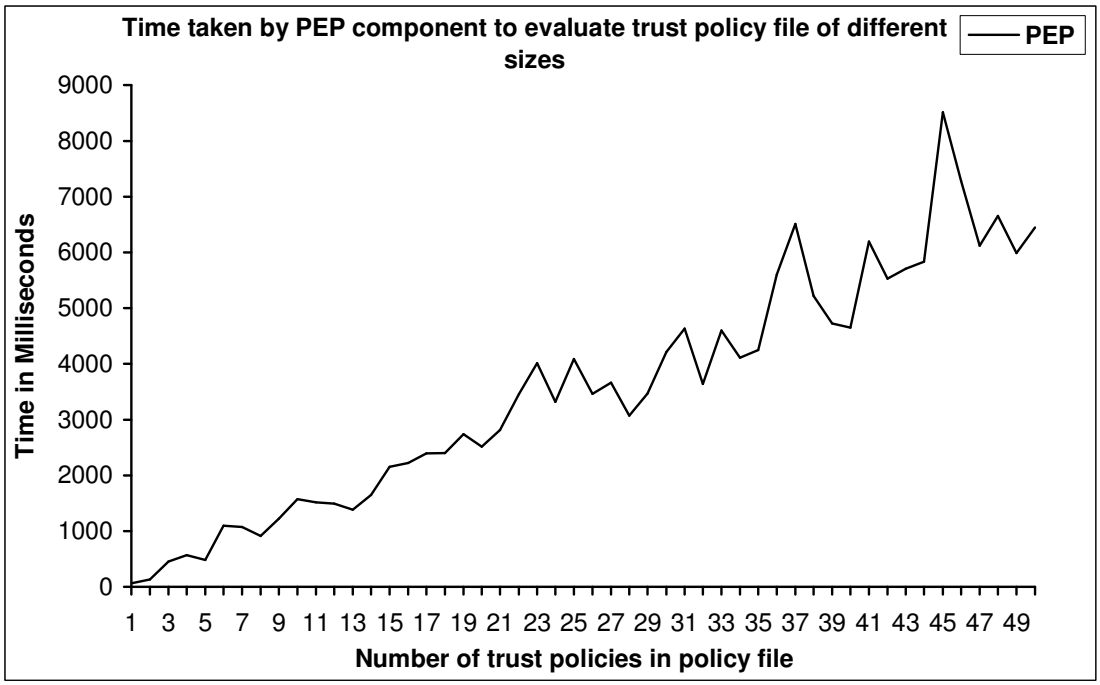

Figure 6.28: PEP time to evaluate trust policy file of different sizes

6.2.4 Performance analysis of authorization policies

The performance analysis of authorization policies involve attaching 50 pure authorization related policies with a service and noting the time taken by PIP, PDP and PEP components for their evaluation individually and accumulatively. These policies are chosen such that they do not involve any authentication, privacy and trust related aspects. The average time taken by PEP component for authorization policies comes out to be 83.58 ms which is significantly less than the average time taken by PEP component for trust policies, and more than that of authentication policies. This value is close to average time taken by PEP component for privacy policies. Also the average time taken by PIP component is 22.55 ms compared to 0.32 ms in case of authentication policies, 24.65 ms in case of privacy policies and 110.74 ms in case of trust policies. The graphs of Figure 6.29, Figure 6.30 and Figure 6.31 show these details.

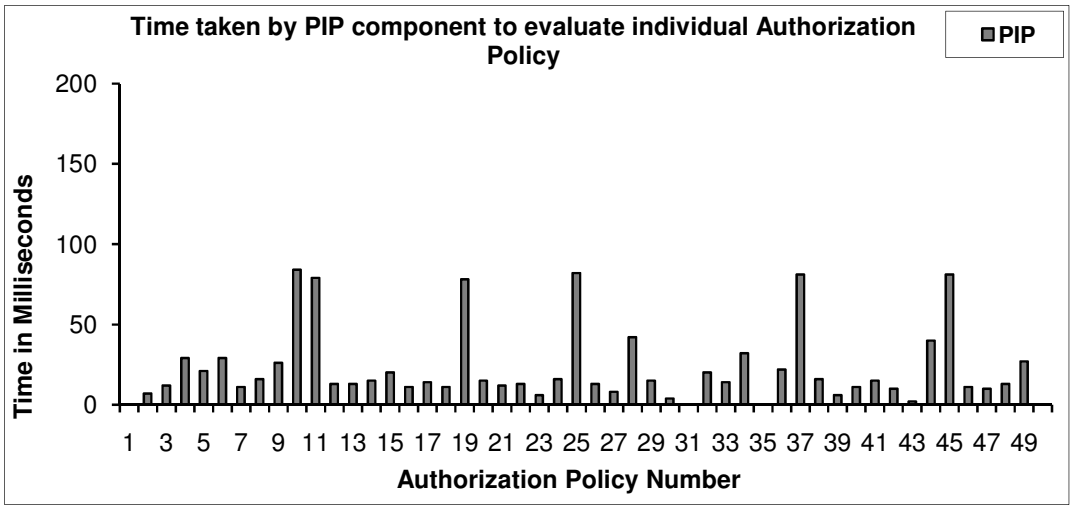

Figure 6.29: PIP time to evaluate individual authorization policy

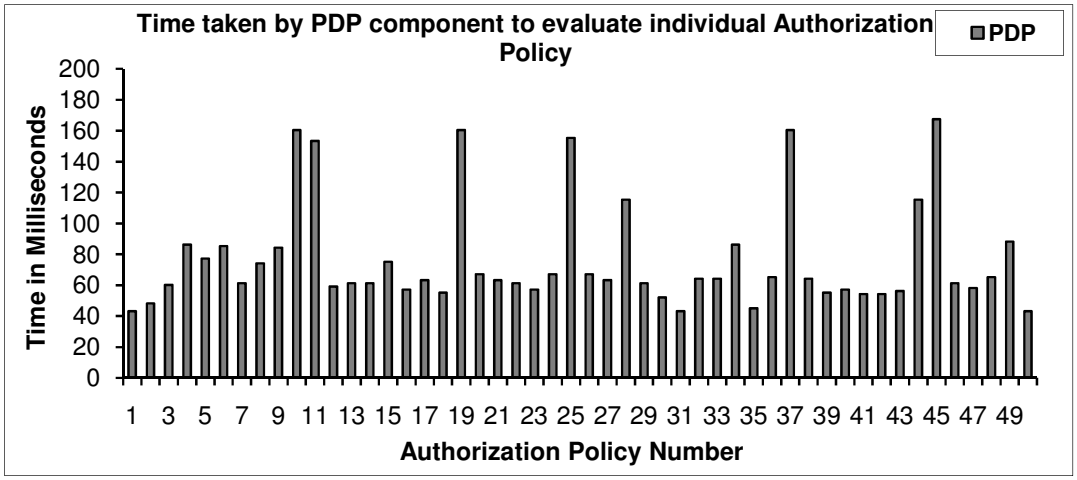

Figure 6.30: PDP time to evaluate individual authorization policy

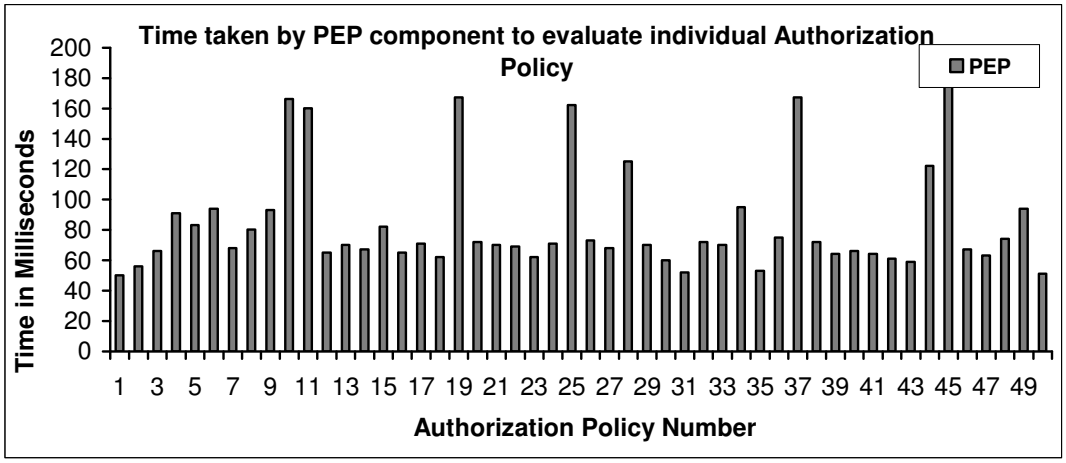

Figure 6.31: PEP time to evaluate individual authorization policy

Figure 6.33 and Figure 6.34 show how the time taken by PIP, PDP and PEP components increases with the increase in number of authorization policies. From these graphs, it is clear that the time taken by all the components increases linearly and not exponentially with the increase in number of authorization policies. Thus increasing the number of authorization policies attached with a service/resource does not adversely affect the performance of the overall framework.

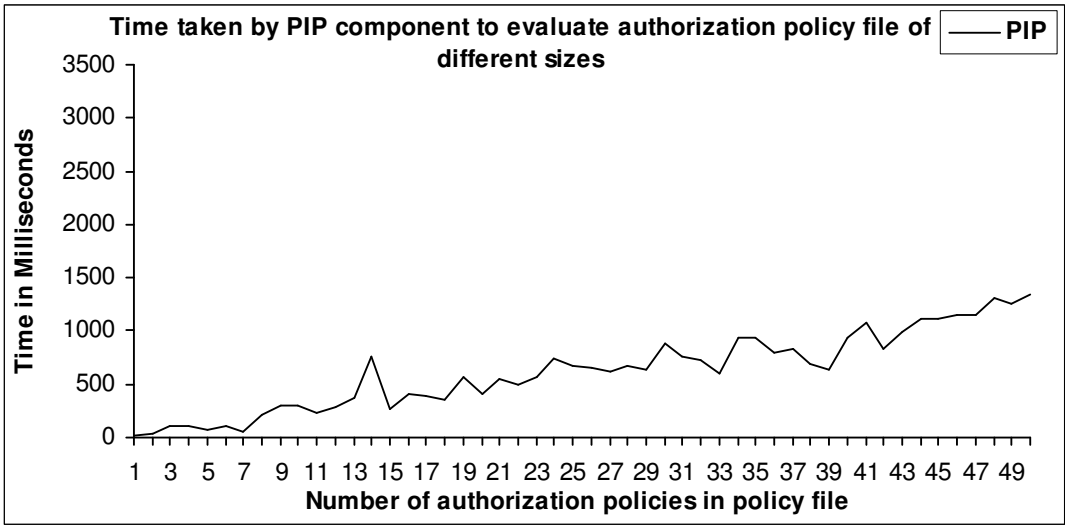

Figure 6.32: PIP time to evaluate authorization policy file of different sizes

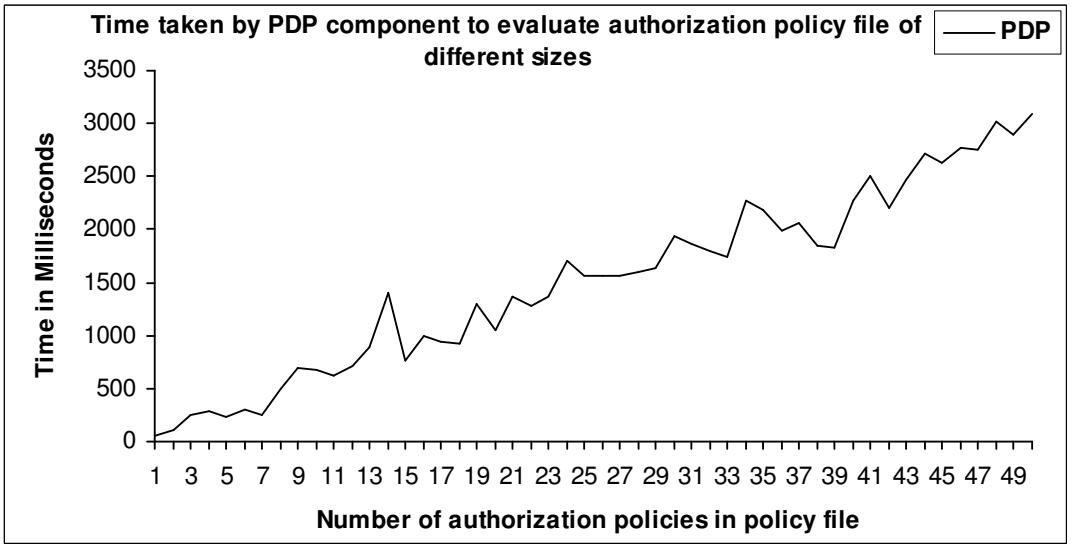

Figure 6.33: PDP time to evaluate authorization policy file of different sizes

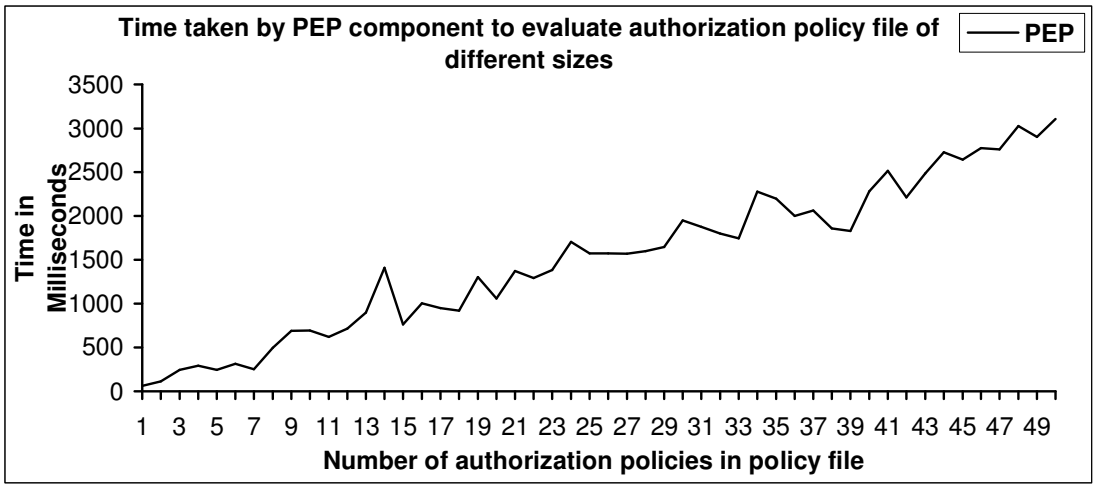

Figure 6.34: PEP time to evaluate authorization policy file of different sizes

6.2.5 Comparison of authentication, privacy, trust and authorization policies

Graphs of Figure 6.35 and Figure 6.36 show the comparison of time taken by PEP component for evaluation of authentication, privacy, trust and authorization policies individually and accumulatively. Graph of Figure 6.35 compares the time taken by PEP component to evaluate individual policy of each type and Figure 6.36 shows the effect of increase in number of policies on time taken by PEP component for all types of policies. Table 6.1 lists the average time taken by PIP, PDP and PEP components to evaluate authentication, privacy, trust and authorization policies.

Time / Policies	Average PIP Time (ms)	Average PDP Time (ms)	Average PEP Time (ms)
Authentication policies	0.3205	46.2665	53.3767
Privacy policies	24.6554	73.5858	81.3369
Trust policies	110.7392	170.0245	178.1562
Authorization policies	22.5524	76.3898	83.5802

Table 6.1: Average time taken by PIP, PEP and PDP components to evaluate authentication, privacy, trust and authorization policies.

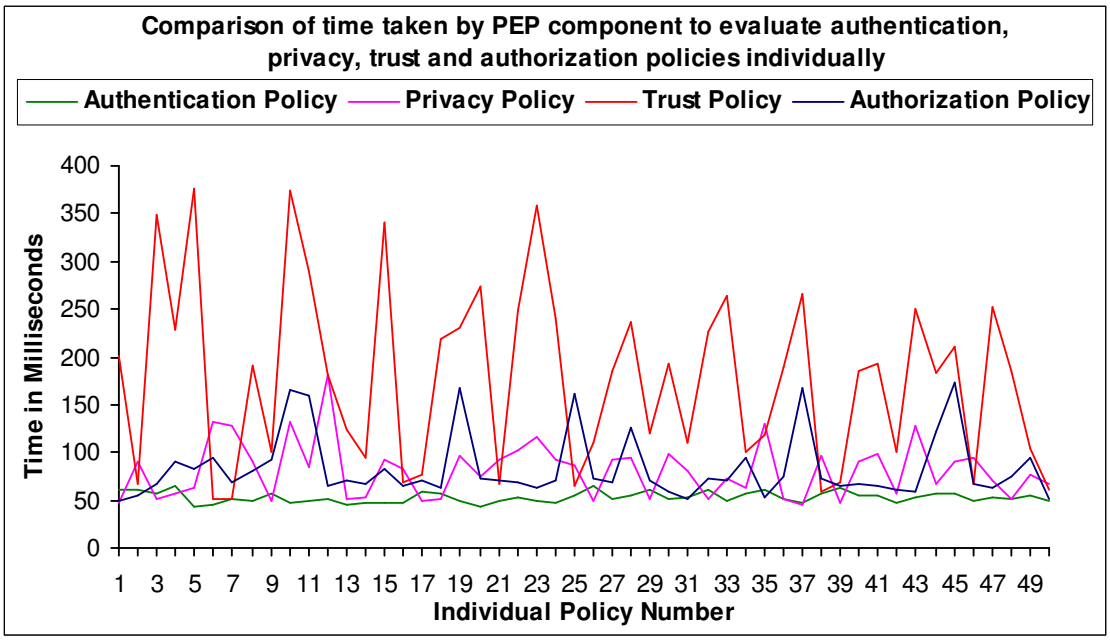

Figure 6.35: Comparison of PEP time to evaluate individual authentication, privacy, trust and authorization policies

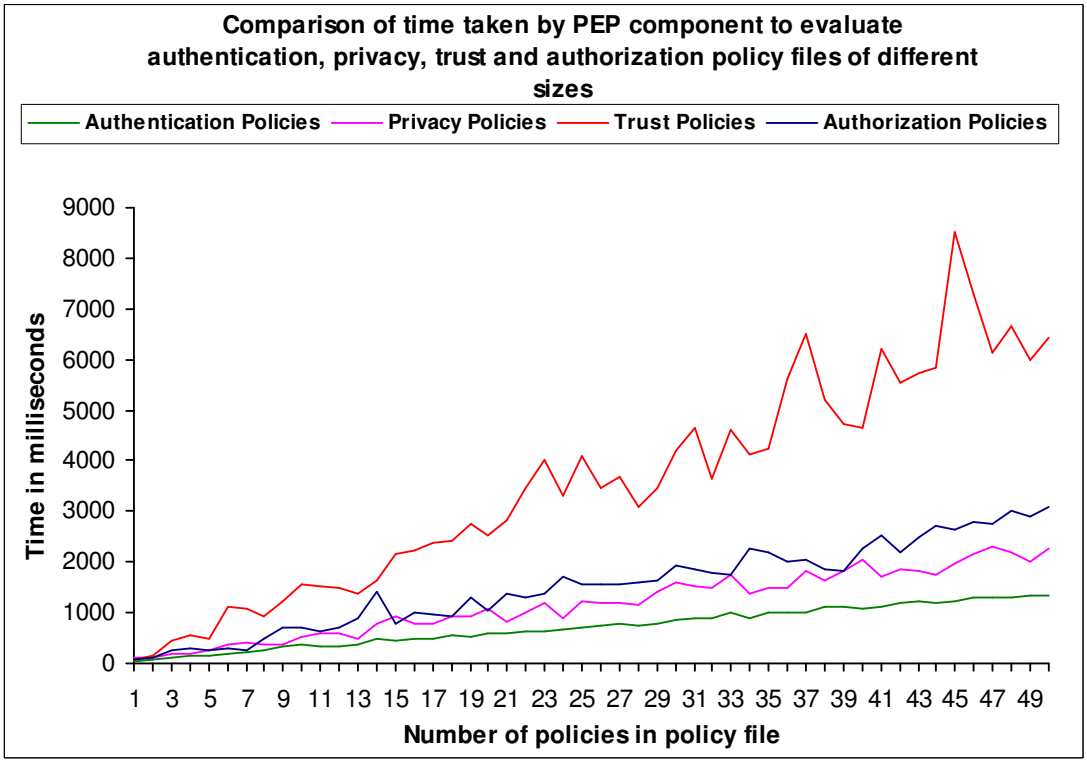

Figure 6.36: Comparison of PEP time to evaluate authentication, privacy, trust and authorization policy files of different sizes

From the graphs shown in Figure 6.35, Figure 6.36 and the values listed in Table 6.1, it is clear that authentication policies take less time and trust policies take more time to evaluate compared to evaluation of privacy and authorization policies. In trust model, the time taken by PEP component increases more assertively compared to other models. This is because of the fact that trust policies involve calculation of trust value (direct as well as recommended) which requires access to history of past interaction taken place between requester and service provider. This is a time consuming process. So trust policies take more time compared to evaluation of authentication, privacy and authorization policies, where no such calculations or access to history of past interactions is required.

Thus performance can be a concern if the number of trust policies attached with a service / resource are more. The time taken by privacy and authorization policies for their evaluation is close to each other. This is because of the fact that the attribute access and evaluation requirements of both types of policies are almost similar. Evaluation of privacy policy mainly involve checking of purpose, allowed actions and privacy conditions whose evaluation is not much different from that of authorization policies. Authentication policies take relatively less time to evaluate compared to all other types of policies because they do not require much access to resource or environment attributes. They mainly require subject attributes and authentication related information which is generally available from the access request itself. The resource and environment attribute access requirements of authentication policies are minimal. Thus they take less time to evaluate compared to all other types of policies.