Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies

Trust Management and Negotiation for Attestation in Trusted Platforms using Web Services

Aarthi Nagarajan, Vijay Varadharajan, Michael Hitchens
Department of Computing, Macquarie University, Sydney, Australia
{aarthi,vijay,michaelh}@ics.mq.edu.au

Abstract

The concept of trusted computing technology is becoming significant in that such technologies are being increasingly available in PCs and mobile devices. With the advent of this technology, one can move from traditional user-only based trust management systems to user and platform-based trust management systems. In this paper, we propose a TCP based trust management and negotiation framework for better security decision making. In this regard, we outline a 3-stage property model that can be leveraged to define policies of different granularities. We then propose how Trust Policy Language (TPL) can be used to create compositions of properties. Finally, the paper discusses the different architectural design choices (such as push, pull and delegation based models) in negotiating trust using these policies and their implications in a distributed web service based environment.

1. Introduction

The notion of 'trust' has played a foundational role in security for quite a long period of time. Dating back to the 70s and early 80s, it was used in the development of Trusted Computer System Evaluation Criteria (TCSEC) [1]. Here "trust" was used in the process of convincing the observers that a system (model, design or implementation) is correct and secure. This was primarily applied to the design of secure operating systems ("trusted systems"), which had the notion of "trusted processes". Later in the 90s, the concept of 'trusted authorities' was familiarized. These authorities are "trusted" to provide guarantees and assurance about the security information such as identities, certificates, roles and services. Then the term 'Trust Management' was introduced which provided a unified framework for specification, management and evaluation of security policies. However, in the design of such secure systems, a high degree of confidence was placed on software-only solutions and trust 'assumptions' were made on the hardware on which these applications were executed. The initiative of the TCG [2] addresses this aspect of hardware security. It provides mechanisms by which a trusted hardware subsystem can provide evidence of behaviour of the platform on which several applications run.

The advent of the trusted computing technology will change the way we reason the trust levels of services available to us. In the paper, we argue the need for a user and platform based trust management system instead of a user only based trust management system. Section 2 discusses about trusted computing platforms and their properties. Section 3 discusses about web services and attestation. Sections 4 and 5 propose a trust management framework for trusted platforms and discuss about the credential and policy aspect of the framework. Section 6 outlines some design issues involved in trust negotiation and finally section 7 concludes with a discussion on future work.

2. Trusted platforms and attestation

The Trusted Computing Group (TCG) has defined a set of open hardware and software specifications for Trusted Computing technology. The heart of the TCG architecture is the Trusted Platform Module or the TPM chip. Using the TPM chip, a Trusted Computing Platform can provide platform authentication, secure data storage and attestation services.

All components of the trusted platform have a Component Validity Certificate (CVC) that vouches for a measurement value corresponding to the component's 'good' working state. When the platform boots, the first program measures itself and passes control to the next measurement agent. A bootstrapping process follows where all the

0-7695-3049-4/07 $25.00 © 2007 IEEE
DOI 10.1109/.78

components of the platform are measured and the measurement values are stored inside the Platform Configuration Registers (PCRS) of the TPM. Since a platform has only limited number of PCRs (e.g. 15 for PCs) but a much larger number of components, many measurements are accommodated into a single PCR by hashing the concatenation of the old value with the new measurement value. The sequence of measurements and hashing are also saved in a measurement log outside the TPM.

Attestation is a mechanism defined by the TCG to report the measured PCR values to a Requestor, who wishes to know the state of the platform to determine whether or not to have an interaction. Using the attestation report, the requestor can verify if the platform measurements are as expected and if the platform is in an acceptable state. However, the use of binary measurements in attestation to report the state of the system can have certain disadvantages. Firstly, binaries do not hide the configuration information of an Attesting Platform (AP) from an Attestation Requestor (AR). This could have some privacy implications for AP. AR could also learn about the vulnerabilities of AP's system and attack it. Secondly, binary values change often. This is because the measurement values of each component change each time the component has a version update or a new patch applied. AR would need to keep updating its security policies as often as the binaries change. Thirdly, binary values are difficult to understand and do not represent well the actual security requirements on a platform as in natural language.

To address these issues of binary attestation, Property Based Attestation was first proposed by [3] and later by [4]. In property based attestation, binary measurements of a platform are mapped to certain security properties satisfied by that platform. Properties are high level statements about a platform, in natural language or a structured variant thereof, describing some aspect of the state of the platform. If properties were used to represent the state of a platform instead of measurements, AR's policies would be easier to understand and manage, and properties may not change as often as hash values do.

3. Web services based attestation

One of the emerging areas in networked computing that is key to businesses and applications is the area of web services. Web services provide a way for evolving the internet into a service-oriented architecture. These services aim to use standard formats and protocols to promote interoperability and extensibility among applications and to enable complex operations. At present, there are several efforts underway that are striving for the provision of security in web services such as authentication between participating entities, confidentiality and integrity of communications (such as WS Security [5]) as well as security policy (such as WS-SecurityPolicy [6]) and trust (such as WS-Trust [7]). WS-Attestation [8] proposes an extension of the attestation architecture in trusted platforms for web services. It defines mechanisms for integrity request and integrity report using web services and proposes extensions to WS-Trust to accommodate the request-response messages. In this section, we will first briefly review the ws-attestation scheme.

Direct Attestation as defined by the Trusted Computing Group is shown in Figure 1 below. Here, a requestor referred to as an Attestation Requestor (AR) challenges a platform referred to as an Attesting Platform (AP) for its state. AP exports a list of PCR values along with the measurement log and the Component Validity Certificates (CVCs) to AR. AR now performs the same computations as AP on the log entries in order to arrive at another set of PCR values. AR asserts AP's platform to be in good state if the computed PCR Values match the sent PCR values and if each of the components' measured values in log matches the corresponding hash value in the CVCs.

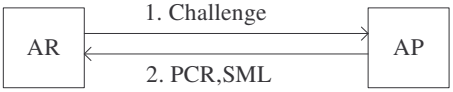


Figure 1: Direct attestation

Given the number of PCRs available in a platform are limited, and the number of components included in one PCR and the number of possible revisions that each software component can go through are large, the log entries can become increasingly rich. Validating the measured values using the log and verifying them against the component validity certificates for every request can be complex for AR. Also, revealing the log information to AR could have certain privacy implications for AP. To address some of these short comings, ws-attestation introduces the notion of a Validation Service (VS). VS is a trusted third party that performs the attestation verification on behalf of AR and provides a signed assertion stating that the platform has certain characteristics (e.g. if all PCR values are validated, 'platform integrity=true'). This signed assertion is called as an attestation credential.

In general, like in any distributed system context, ws-attestation also proposes three different architectural models for distributing the attestation credential between the parties. Figure 2 shows a pull model where AR presents a challenge to AP and AP responds to the challenge with its state information (PCR values, log and validity certificates). AR now requests VS to perform the validation of AP's state on its behalf. VS verifies the attestation report and generates an attestation credential for AP. Figure 3 shows a push model where AP requests VS to validate its platform and issue a credential in advance. When AR presents the challenge, AP provides this credential to vouch for the state of its platform. Finally, in the delegation model as shown in Figure 4, AR delegates VS to provide it with AP's attestation credential at the time of the challenge.

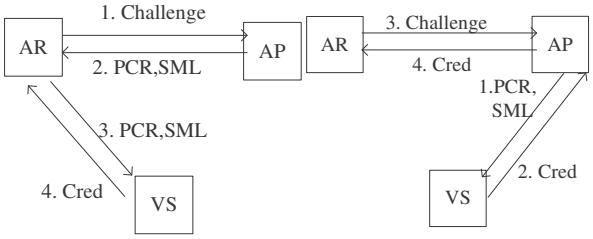


Figure 2: Pull model

Figure 3: Push model

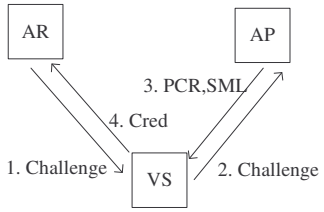


Figure 4: Delegated model

4. Distributed trust management and negotiation for trusted platforms

In distributed systems like web services, it often happens that the service provider and the service requestor are strangers and do not have a pre-established trust relationship. Therefore, at the time of service request, a requestor may prove to the provider that it possesses the necessary credentials to be able to access the service. The provider may then decide whether or not such credentials are sufficient to service the request and act accordingly.

The term 'Trust Management' was first coined by Blaze at al. in [9]. A trust management system provides a unified framework for the specification of security requirements of a service provider in the form of policies, specification and distribution of credentials required by requestors to satisfy these policies and a mechanism to interpret if a set of available credentials is sufficient to authorize a requested service. Also, if the credentials provided by the requestor are inadequate, it must be possible for the service provider to request for more credentials such that a sufficient trust stage is established to provide the requested service. This process by which the communicating entities request for and exchange more credentials is known as 'trust negotiation' [10].

In a traditional authorization system, when one principal requests a service from another, the receiving principal needs to address at least two questions. Is the requesting principal the one it claims to be and does the requesting principal have the appropriate privileges for the requested service? These questions relate to the user authentication and user authorization requirements of a system. Using trusted computing technology, we can see how we can move from user-only based authorization systems to user and platform-based authorization systems. A requestor R presents its user credentials and platform credentials along with the request. The Provider P now needs to answer two additional questions; is the requestor's platform what it claims to be and does the requestor's platform have the necessary properties to allow access?. The provider may choose to determine whether the state of the client system is acceptable for it to provide the service even before deciding whether to provide the requested service or not and before going through the authorization process. If the provider finds the requestor's state to be unsuitable based on its platform related policy, then it can decide not to go through the user related policy evaluation process. Hence we can see that traditional distributed authorization services along with trusted functionality and properties provided by the trusted platform via its TPM, can lead to better decision making [11].

Therefore, a holistic trust management and negotiation framework for trusted platforms requires certain important elements. Firstly, there is the need for a policy language to capture the user and platform based requirements of a service provider as policies. Secondly, user and platform based credentials are required to capture the available privileges of a service requestor. Thirdly, a negotiation protocol that defines how a negotiation can be initiated in the context of attestation and finally an evaluation algorithm to determine if the available credentials suffice the policy requirements to service the request. The rest of the

section discusses about the platform-based credential and policy aspect of the trust management system. Section 6 discusses about the trust negotiation design choices available for such a system.

4.1 Credentials for trusted platforms

Authentication Credentials are used to provide proof of identity. Presently, a TCP can be identified using either an Endorsement Key (EK) credential or an Attestation Identity Key (AIK) credential. Because the EK credential can be used only for taking ownership and for creating other AIK credentials, it cannot be used to identify the platform. The AIK on the other hand, is public-private key pair that is generated inside the TPM. A privacy CA attests the public part of the AIK and issues a signed AIK credential. The private part of AIK never leaves the TPM and is used to sign messages generated by the platform. Each trusted computing platform can have many AIK credentials, typically one for each user of the platform. Requestors can provide their AIK credentials to service providers in order to prove the authenticity of their platform.

Authorization credentials for platforms are more complex than authentication credentials because authorization credentials should capture platform based authorization requirements. This information could include platform identities, platform properties, component identities of a platform and properties satisfied by each of the component. We think it is important to capture properties of components at different granularities. Presently, we propose a 3-level granularity model with levels, high, mid and low. A high level property expresses a property of a platform as a whole, where a platform is composed of different hardware and software components. These properties are less expressible but provide more privacy for the attesting party. A low level property expresses a property of an individual component of the platform. These properties are highly expressible but provide very little privacy for the provider. A mid level property is at a level of abstraction between the high and low level properties. We group many low level properties into a bucket and assign the bucket a mid level property label. A component can be assigned to a mid level property label if it satisfies one or more of the low level properties with in the label's bucket. Such policies can be moderately expressible and at the same time provide a certain level of privacy for AP. This is because, availability of a mid level label does not disclose information about the specific properties satisfied within the set.

We introduce the notion of a Component Property Certificate (CPC) as a platform credential that is capable of capturing such properties of different granularities. Each component of a platform can be issued a CPC either by the component manufacturer or by a trusted authority vouching for certain properties of the component. A CPC could essentially be made of the elements <ComponentID, Expiry Date, Property Set, Issuer Signature> elements. The property set can include different properties of different granularities.

4.1.1. Policy types based on property types

Based on the properties of components that can be expressed at three levels of granularities, a service provider can capture these property requirements as policies. We propose a three stage granularity policy model as shown in Figure 5.

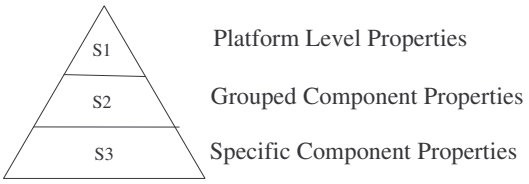


Figure 5: Three stages of platform based polices

At stage one S1, authorization policies are at based on the high level properties of the platform. For example, AR considers the platform to have 'platform integrity property' if all the components of the platform are integrity protected. This means that for all the components measured in the platform, the measured values are consistent with the expected values in their component validity certificates. Such policies do not require property details of individual components and provide more privacy for the attestation provider (AP). As there are fewer such properties, they are represented on the top of the pyramid.

At stage three S3, policies are extremely fine grained and are based on the low level properties of the components. Policies are based on every individual component of the platform and properties of components that should be satisfied. Although such properties of components may be difficult to evaluate, such policies can provide a high level of flexibility in expressing the requirements.

Stage two S2 polices are neither very abstract nor very fine grained. S2 policies are drawn from the mid level properties and provide a level of abstraction that is between S1 and S3 policies. AR defines policies based on the labels that the components are expected to have.

Since security labels directly map to security properties, AR can believe that components with certain labels satisfy one or more properties that are defined with in the label's bucket. Such middle level policies can be reasonable to define, provide sufficient privacy for the attesting party and sufficient flexibility to express policies based on properties.

5. Policy specification for trusted platforms

In the previous sections, we have outlined the credential aspect of the trust management system for trusted platforms. We defined the structure of platform credentials that can be used in evaluating authentication and authorization policies, and described the different stages of policies that can be defined based on policy granularity. In this section we will discuss the need for compositions and how the Trust Policy Language (TPL) [12] can be leveraged to generate composition credentials for trusted platforms.

Types of policies chosen are clearly determined by various factors. An attestation requestor can choose to have different granularities of policies for different services requested or for different service requestors. Alternatively, granularity can also depend on platforms that request the service or even the different components that make up the platform. Therefore, while expressing policies for platforms and assigning them to permissions, it is important that policies accommodate different types of platforms, different components of the platform and different types of properties of components.

```
<GROUP NAME = "Partner Employee">
   <RULE>
        <INCLUSION ID= "vemp" TYPE= "Partner"
        FROM= "My Employee" REPEAT= "2"></INCLUSION>
          <FUNCTION>
                <GT>
                    <FIELD ID= "vemp" NAME= "level"></FIELD>
                    </CONST>3</CONST>
                </GT>
            </FUNCTION>
   </RULE>
</GROUP>
```

Figure 5: TPL policy specification for role based access control

IBM's Trust Policy Language (TPL) [12] is a language that is used to define the mapping of strangers to predefined business groups based on certificates issued by third parties. These certificates may be identity certificates like X.509 and the issuers of the certificates are either known in advance or provide sufficient proof to be considered trustworthy. TPL is an XML based policy language. At the highest level, there are *groups* and under each group, there are *rules* for group membership. Each rule has some inclusion statements to define valid certificates that it should check and a function to define conditions on certificate fields. We refer the readers to [12] for a better understanding on TPL. A simple example is shown in Figure 5.

According to this policy, for a requestor to become a part of the group *Partner Employee*, he needs to provide two partner type certificates signed by existing members of *My Employees* group and the level of the employee field in the certificates should be greater than 3. If the group represents roles, then a role based access control system can use this policy language to map strangers to *Partner Employee* roles. It should be noted that TPL does not specify policies for what a member of the *Partner Employee* role can do.

Similarly, in a trusted platform, it is likely that policies are based on a large number of components. Each of these components may have properties that may change during the life time of the component. Given the multitude of the number of components in a platform, the possible revisions each of these components can have, and the properties that may change accordingly, policies would need to be updated regularly to reflect the changes. Allowing requirements to be composed into *Compositions* means that these compositions can be assigned to different permissions. We emulate the model of RBAC [13] where users are assigned to roles and roles are assigned to permissions. When users change with in an organization, only the user-role relationship changes and the role-permission assignments are not affected. Similarly, we assign permissions to compositions. When components or properties or property types of components change, only the composition is updated leaving the mapping between the composition and permissions unaltered. Just like a role credential can be mapped to a role using TPL, we map a platform authorization credential or Component Property Certificate (CPC) to compositions using TPL as shown in Figures 3 and 4.

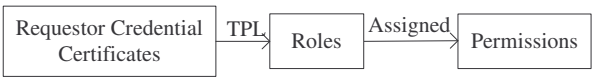


Figure 6: Mapping requestor credential certificates to permissions

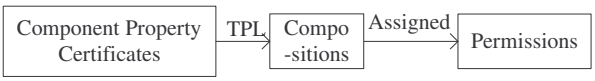


Figure 7: Mapping component property certificates to permissions

In TPL, the <GROUP> elements make the highest level. Under each group are <RULES> elements for group membership. Each rule has some <INCLUSION> statements to define the conditions for <GROUP> membership. A <FUNCTION> within a rule defines conditions for inclusion. For Composition Based Access Control (CBAC), we use TPL to check if a set of CPCs can be assigned to a certain group. <RULES> can define properties that should be available in a CPC in order for it to be allocated to that group. <INCLUSION ID> is the identifier of a CPC for which functions are written and the <INCLUSION TYPE> can indicate that the certificate is of type CPC.

```
<GROUP NAME =“Acceptable Browser">
  <OR>
    <RULE>
      <INCLUSION ID="ie" TYPE=“CPC"> </INCLUSION>
       <FUNCTION>
         <AND>
           <EQ>
             <FIELD ID="ie" NAME="name"></FIELD>
             <CONST>internet explorer</CONST> </EQ>
           <GT>
             <FIELD ID="ie" NAME="version">
             <CONST>6</CONST> </GT>
       </FUNCTION> </RULE>
    <RULE>
      <INCLUSION ID="ff" TYPE=“CPC"> </INCLUSION>
      <FUNCTION>
        <AND>
          <EQ>
            <FIELD ID="ff" NAME="name"></FIELD>
            <CONST>firefox</CONST> </EQ>
          <GT>
            <FIELD ID="ff" NAME="version">
            <CONST>1.5</CONST> </GT>
      </FUNCTION> </RULE>
  </OR>
</GROUP>
```

Figure 8: TPL policy specification for composition based access control

Figure 8 shows an instance of mapping CPCs to groups. In this example, a bank defines its requirements on a platform's browser component in order to allow internet banking. The browser can be assigned to a group 'A*cceptable Browser'* only if it has the identity 'internet explorer' with version greater than 6 or if it has the identity 'firefox' with version greater than 1.5. This policy can also include other properties expected in a CPC such as the support for scripts, phishing filter, 128-bit encryption etc and their property type (e.g. S3). Then this composition 'Acceptable Browser' can be assigned to permission 'allow internet banking'.

In this paper we do not discuss how permissions are assigned to compositions and how these policies can be combined with user based policies. It is possible to extend any policy language for web services as XACML [14] for this purpose, which we are currently working on. Such a policy language could evaluate if a requestor assigned to a role R whose platform belongs to a composition C is allowed to access the requested service S.

6. Trust negotiation in ws-attestation

In the previous sections we defined the credential and policy aspect of a trust management system using trusted platforms. We also illustrated the need for compositions and how compositions can be assigned to permissions using TPL. In this section, we will present the design choices available for trust negotiation between AP and AR. We illustrate this using the push model as an instance and outline how trust can be negotiated in the three policy stages S1, S2 and S3. It would be fairly similar to extend the same protocol for the push and the delegated models as well.

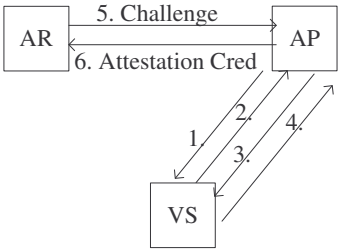
Let us assume that the Attestation Requestor (AR) is the service provider and Attesting Platform (AP) is the service requestor using web services. Let AR have a policy base with all three stages of policies S1, S2 and S3 defined depending on the type of request. AP has user credentials for user authentication and authorization. It has a platform AIK credential for platform authentication and all the necessary component property certificates for authorization. AP requests a service from AR and AR challenges AP for the state of its platform. We also assume that AR trusts all the trusted authorities that issued AP its AIK and CPC credentials and VS that will validate the PCR Values, log and validity certificates for AP.

6.1. Trust negotiation for S1 policies

S1 policies are platform level policies and are limited in scope. Attestation credentials issued by VS as discussed in section 3 can be used to satisfy S1 type of policies. In the push model, when AP requests an attestation credential, the Validation Service (VS) verifies all the PCR values of the platform. It then issues an attestation credential validating one or more properties (e.g. 'platform integrity is 'true').

In this case, if all the PCR values are valid, a positive credential is issued with the property value as 'true'. Otherwise, VS initiates a negotiation (as shown in Figure 9) with AP by making a log of those PCR values that did not verify, sending the log to AP and requesting AP to take the necessary actions to make

the corresponding PCR values valid. VS may not issue a positive credential as long as all the required PCR values are not completely validated. When AR challenges AP for the state of the platform, AP presents this attestation credential. (AR can request AP again for a most recent credential with the required properties, if it believes that the credential does not represent the most recent state of AP). This trust negotiation model can also be extended to the pull and the delegated models.



1. Request 1 -PCR,SML
2. Log of invalid PCRs
3. Request 2 -New PCR,SML
4. Attestation Cred

Figure 9 : Trust negotiation for S1 policy type in push model

6.2. Trust negotiation for S2 policies

Now let us assume that AR has S2 type of authorization polices defined for the requestor to access the requested service. S2 type of policies has permissions assigned to compositions and every composition is made of components with certain security labels. In the push model, when AP sends the attestation credential to AR, AR evaluates it with its policy base and understands that AP should be assigned to a composition in order to provide the requested service. AR initiates a negotiation and requests AP for the Component Property Certificates. AP can now send all its Component Property Certificates to AR with security labels in the 'property set' field'. AR evaluates the CPCs using TPL and determines if AP can be assigned to the required composition. If the necessary security labels are unavailable, it can request AP for more CPCs. As shown in Figure 13, after receiving the credential and CPCs in step4, AR performs the policy evaluation and based on the result, accepts or denies the request from AP.

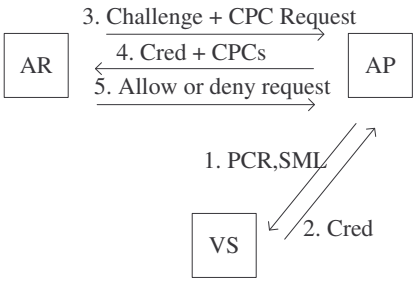


Figure 10: Trust negotiation for S2 policy type with evaluation at AR

Alternatively, AR can make its policy available to the VS to perform the policy evaluation and let AR know if all the components of AP can be assigned to a composition or not. This is shown in Figure 14. When AR presents the challenge to AP, it also makes a request for its CPCs as shown in step 3. AP responds with the attestation credential and the CPCs. AR then requests VS to perform policy evaluation on its behalf as shown in step 5. VS evaluates the CPCs against the TPL policy and sends the result to AR if the components can be assigned to a composition or not. However, it is the design choice of AR to determine if AR wishes to perform the policy evaluation itself or push the policies to VS or make the policies available public policy base for VS to do the evaluation on its behalf.

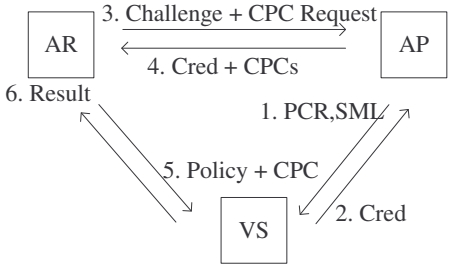


Figure 11: Trust negotiation for S3 policy type with evaluation at VS

6.3. Trust negotiation for S3 policies

Now let us consider that AR has S3 type of policies defined for the requestor AR for the requested service. When AP sends the attestation credential to AR, AR compares it with its policy base and finds that it requires specific components of AP to have certain properties such that these properties can be mapped to a composition. AR initiates a negotiation with AP. The negotiation protocol for S3 policy is same as that of S2 policy type. However, the difference is that S3 policies are extremely fine grained and are based on the individual components' properties unlike the S2

policies which are based on middle level labels. AR and AP can agree on what S3 properties are required for the service and what can be disclosed during negotiation. When AP receives the required credentials, it can do the policy evaluation. It is again a design choice for AR to determine who should do the negotiation (VS or AR) and who should do the policy evaluation (VS or AR) and how the policies will be available to VS if required.

7. Conclusion

The concept of trusted platform using the trusted computing technology such as the TPM is becoming significant in that such technologies being increasingly available in PCs and mobile devices. One of the key design issues is the ability to represent trust policies and make use of them in decision making when such trusted platforms are used in applications. This paper has made some contributions which we believe are an important first step in achieving trust policy based decision making. In this paper, we have outlined an architectural framework for specification and negotiation of trust policies between parties using the trusted platform technology. In this context, we have argued for the need for a higher level abstraction that is able to capture the low level state of the trusted platform (in terms of attributes such as the PCR values) use this in the evaluation and negotiation of trust between the communicating entities. We have introduced the concept of Component Property Certificates, which can be used in specifying and building trust relationships. Then we have described the different levels of polices that need to be specified and negotiated. In this regard, we have identified platform integrity, generic component and specific component type trust policies. Then we have examined the different architectural design choices (such as push, pull and delegation based models) in negotiating trust using these policies and their implications in a distributed web service based environment. There are several areas of further work that can be pursued.

We are currently in the process of refining the notion of component property certificates and defining the labels and property set that can be specified. We are also considering extensions to the TPL, XACML and WS-Trust languages to specify the above trust policies. Then we plan to use such trust policy based secure decision making in the design of applications such as peer to peer computing and grid computing.

8. References

[1] “Trusted Computer System Evaluation Criteria (TCSEC)", Dept of Defense, 1985.

[2] "TCG TPM main specification version 1.1b", Trusted Computing Group, 2004.

[3] Poritz, J., et al., "Property attestation-scalable and privacy-friendly security assessment of peer computers", Technical report RZ 3548, IBM Research, 2004.

[4] Sadeghi, A.-R. and C. Stueble. "Property-based attestation for computing platforms: Caring about properties, not mechanisms", New Security Paradigm Workshop (NSPW), 2004.

[5] "Web Services Security: SOAP message security 1.1", C. Kaler, Editor, OASIS standard specification, 2006.

[6] "Web Services Policy Framework (WS-Policy)", Version 1.2, W3C, 2006.

[7] "Web Services Trust Language (WS-Trust)",W3C,2005.

[8] Yoshihama, S., et al., "WS-Attestation: Efficient and fine-grained remote attestation on web services", IBM Research, 2005.

[9] Blaze, M., J. Feigenbaum, and J. Lacy, "Decentralized trust management, in Security and Privacy", 1996. p. 164-173.

[10] Yu, T., M. Winslett, and K.E. Seamons, "Interoperable strategies in automated trust negotiation", 8th ACM conference on Computer and Communications Security, Philadelphia, PA, USA, 2001.

[11] Varadharajan, V., "Authorization and trust enhanced security for distributed applications", Seventh International Conference on Information and Communications Security (ICICS), Beijing, China, 2005.

[12] Herzberg, A., et al., "Access control meets public key infrastructure, or: assigning roles to strangers", IEEE Symposium on Security and Privacy, 2000.

[13] Sandhu., R.S., et al., "Role-based access control models", *Computer*, 1996, 29(2): p. 38-47.

[14] "eXtensible Access Control Markup Language 3 (XACML) Version 2.0", OASIS, 2007.

[15] Nagarajan, A., Varadharajan, V. and Hitchens, M. “Trust management for trusted computing platforms in web services”, (to be presented) STC, Virginia, USA, 2007.