Secure Hardware-based Distributed Authorisation Underpinning a Web Service Framework

Marco Casassa Mont, Adrian Baldwin, Joe Pato
Trusted Systems Laboratory
HP Laboratories Bristol
HPL-2003-144
July 17th , 2003*

E-mail: marco_casassa-mont@hp.com, adrian_baldwin@hp.com, joe.pato@hp.com

authorization, access control, secure hardware, web services

This paper presents a distributed authorisation model suitable for use in a web service framework where multiple parties are involved in performing a particular transaction. The authorisation model uses a third party authorisation service that checks users or services' credentials against a set of authorisation policies. A traditional service provision model does not scale well for such transactions. The proposed model uses a hardware security appliance to deliver the service to the most appropriate site involved in the transaction. The authorisation model supports a multi-party session so that authorisation policies can be checked and built as part of the web service composition process.

* Internal Accession Date Only
Approved for External Publication

Secure Hardware-based Distributed Authorisation Underpinning a Web Service Framework

Marco Casassa Mont (marco_casassa-mont@hp.com)
Adrian Baldwin (adrian_baldwin@hp.com)
Joe Pato (joe.pato@hp.com)

Trusted Systems Laboratory
Hewlett-Packard Laboratories

Abstract

This paper presents a distributed authorisation model suitable for use in a web service framework where multiple parties are involved in performing a particular transaction. The authorisation model uses a third party authorisation service that checks users or services' credentials against a set of authorisation policies. A traditional service provision model does not scale well for such transactions. The proposed model uses a hardware security appliance to deliver the service to the most appropriate site involved in the transaction. The authorisation model supports a multi-party session so that authorisation policies can be checked and built as part of the web service composition process.

1 Introduction

Web services offer the ability to form complex and ad hoc services involving multiple parties. This opens up a large number of security issues. In this paper, we address one particular issue – that of authorisation within a web service framework. In particular, we propose an authorisation model, which allows for complex authorisation polices whilst ensuring trust and privacy between the services and a client as well as ensuring the solution is efficient and scalable.

The proposed authorisation model relies on creating a separate authorisation domain where authorisation decisions for a complex (multi-party) web service transaction are made. In many situations, a trusted third party would provide this authorisation domain. Rather than following a simple third party authorisation service model an alternative service delivery model based on secure hardware is used. This model allows a separate authorisation domain to exist using a secure hardware-based service. It can be located at any point within the service provision chain – the hardware protects the service from subversion by those who have physical access to the device. This leads to an authorisation model that allows computation to occur at the edge of a transaction whilst maintaining the privacy of all authorisation information.

This paper starts with a brief outline of web service frameworks and the way services may be composed into larger services. Section 3 then describes authorisation problems associated with web services, both stressing the need to separate authorisation policies from the normal business functionality and the problems exhibited when multiparty services are used. The need for a separate authorisation domain is then introduced. The proposed authorisation model is then described in the next two sections. Section 4 explains how an authorisation domain or service is created and delivered using a *hardware security appliance* (HSA). Section 5 describes how HSA based authorisation services solve the authorisation problems associated with web services. The model naturally provides strong privacy enforcement for authorisation policies and personal data used in the authorisation process. We proceed to explain how our model avoids server bottlenecks thereby addressing computational efficiency

issues associated with third party authorisation systems. The final discussion addresses some of the properties and wider issues relating to the model.

2 Web Services

Web service frameworks, including Microsoft .Net [1] and Java-based [2], offer ways of delivering complex services over the Internet. These frameworks support user interactions through their web browsers and *html* form interfaces as well as programmatic interfaces through SOAP procedure calls. These communication mechanisms allow a range of services to be offered over the Internet from simple client/server applications through multi-party services built using a number of individual web services. This section, very briefly, sets out how web services can be used hence grounding the authorisation model proposed in this paper.

A web service framework provides a mechanism for simple client/server interactions, for example, an e-commerce site with clients using a browser. More importantly, they provide a framework for allowing collaborations between backend IT systems from multiple companies through programmatic interfaces. They are starting to be used for a range of scenarios, from supporting collaborative projects through optimising supply-chain management. These types of scenarios obviously have some critical authorisation issues. This paper proposes an efficient authorisation model addressing these issues and that preserves privacy between the various parties.

This paper uses a hypothetical travel service scenario to illustrate authorisation issues and describe how the proposed model works. In this scenario, a user, perhaps within a company, goes to a web portal to choose a travel agent. The user would then contact this on-line travel provider to book their journey. The travel request may involve booking plane tickets, a hotel and a hire car each through different component web services. The travel service may be programmed to choose from a set of preferred partner services or alternatively web services could be found dynamically through a market place.

3 Web Service Authorisation

Much of the web service research concentrates on how to provide service functionality, and how to compose the services. Many security issues must be addressed before web service frameworks can be easily used in business critical situations. This section discusses issues relating to authorisation within a web service framework as a precursor to setting out an authorisation model (sections 4 and 5).

Many web services use a very simple access control model based on placing access control lists on particular web pages. Where there is code behind these services performing tasks at the business level, authorisation mechanisms will often be directly coded in the application. This produces maintenance problems. More advanced and flexible solutions separate such rules from the application into an authorisation server, for example as described by [3]. Throughout this paper, these rules are described as authorisation policies – the form of these rules is not significant. They may be arbitrarily complex but the resulting decision must be computable using information about the user and their rights.

This simple form of web service keeps all the authorisation functions within each web service's boundary; each will be responsible for maintaining users' accounts, a credential or rights database as well as maintaining and evaluating authorisation policies. Over time, it is believed that there will be a number of credential providers each asserting that particular individuals have particular properties (e.g. attributes, rights, roles, job titles, credit limits, nationalities). These credential providers form trust services [4] that specialise in validating certain attributes and issuing and managing credentials [5] [6] for these attributes.

This type of credential provision helps distribute the authorisation process; or the process of managing rights and attributes leading to authorisation architectures such as that shown in

Figure 1. In these types of architectures, some credential provision may be distributed and the local service provider may maintain some user information. The authorisation system is still tightly bound to the web service infrastructure and, where complex authorisation policies are involved, the authorisation engine can become a bottleneck.

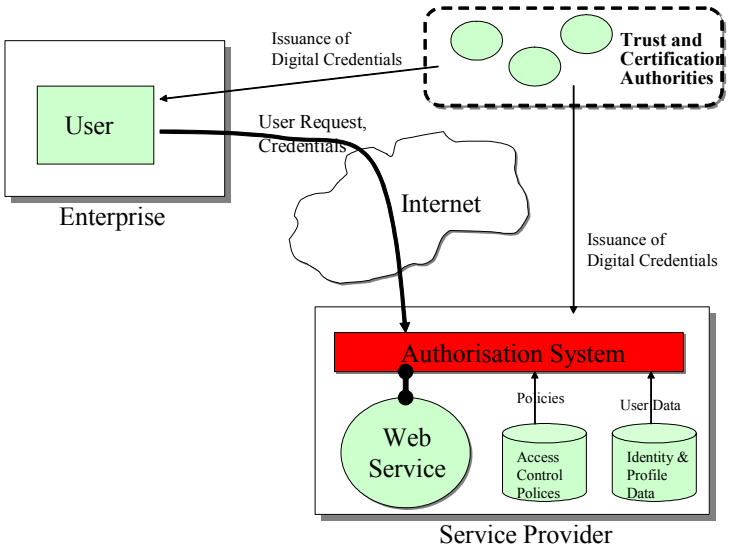

Figure 1: A traditional web service architecture

This can be illustrated with an example; let us consider a scenario where a corporate purchaser makes a request to buy new computers for say £5000. They would accompany their request with their corporate credentials showing they are an employee of that company. Before enacting the web service, the request is filtered through the authorisation system. The system first finds the rule or policy associated with the request and then obtains any supporting information such as the company account information or the user's credit card payment credential. The policy is then checked to ensure that the transaction with that person, company and payment information is acceptable; only valid requests are passed through to the web service.

Research has been done on distributed authorisation mechanisms, including [7] [8] [9] [10] [11], and related systems to address the efficiency problem and reduce bottlenecks. Most of the proposed systems are purely based on a *software approach*. They scale within enterprises or organization boundaries but fail to address trust, privacy and security issues in broader contexts, such as B2B environments. Software solutions are often ad-hoc and heavily reliant on PKI infrastructures and consequentially inherit limitations due to scalability, management and trust issues related to such infrastructures.

3.1 Authorisation and Composed Services

Web service frameworks present opportunities for considerably more complex interaction models than the simple client/server situations discussed previously. These range from more symmetric peer-to-peer transactions, where both parties want to check the others authorisations, through to multi-party composed services, where authorisation is an issue for each component service.

Using the example of the on-line travel reservation service, there are a number of authorisation requirements. The user might need to check the travel provider meets their (or the corporate[1]) policies. This introduces the need for a two-way authorisation process. During

[1] As we develop our example, the inclusion of corporate policies becomes more interesting since these are controls that the user is not necessarily aware of and can't compensate for in direct manipulation of the service client.

the booking processes, the travel agent will need to check their authorisation policies are met; for example, by checking the user's identity, payment and passport credentials. Other composite services will also require that the user match their authorisation policies; for example, on booking a plane ticket the airline would need to show they have an appropriate passport and visa credentials.

The above example suggests a number of authorisation issues that are problematic for conventional authorisation services. An appropriate authorisation framework is needed to allow for the smooth flow of a transaction between multiple parties whilst respecting the privacy of the personal data used in satisfying the policies. The rest of this section describes problems occurring in general web-service authorisation, which are then addressed in sections 4 and 5, where an alternative authorisation framework is proposed.

- *Client validating proposed services.* As well as the service provider checking the client, the client may need the ability to check the service provider matches their authorisation policies associated with such service provision. Some interactions may not take the form of client/server but may involve a negotiation between two peers and in this case, the authorisation checks are symmetric and carried out by both entities.
- *Each composed service has unique authorisation policies*. Services may be composed from multiple services and each individual service may have its own authorisation requirements. The conventional authorisation service discussed previously does not aid this type of transactions where, for example, a coordinating service would need to exchange policy and credential information as well as managing the transaction details. Managing these authorisation exchanges can lead to processing bottlenecks within the service as well as privacy concerns given that the co-ordinating service retains visibility and control.

There are two extremes within composed service frameworks: pre-composed and dynamically composed services. Both have particular authorisation issues. A composed service may be formed from well worked out relationships – here services all have existing trust and business relationships and may share authorisation information (at least in a controlled manner). Alternatively, services may be composed on the fly to meet particular requests – for example finding a hotel at a particular location. This suggests a need to communicate authorisation information between the services and clients.

3.2 Authorisation Domain

Separating the authorisation aspects from the general service functionality, into a set of rules or policies that can be run in a separate authorisation service, aids the manageability of web services. The service provider needs to define and maintain these authorisation policies i.e. under what conditions a given transaction is acceptable. They must enforce the policies before allowing access to a service but they *do not* need to make the policy decisions themselves – this can be done in a separate (authorisation) domain if they trust this domain.

Figure 2 shows this type of situation where the web service is provided in one domain where the policies are also defined and maintained. Within this service domain, there is also a policy enforcement point (PEP) ensuring services are only provided to those having the appropriate rights. The authorisation service, that is the policy decision point (PDP), is now in a second domain; the user is in a third domain. The request must pass through the authorisation domain before reaching the web service. The authorisation domain will produce an appropriate ticket that can be trusted by the policy enforcement point.

From the web services point of view it is critical that the authorisation service is trustworthy and cannot be influenced or subverted by the user. The web service will probably also be concerned about the privacy of their policies and the requests that they are receiving. They must trust the authorisation domain – they are allowing them to see their requests and

credentials. A separate authorisation domain could be created as a trusted third party service and either the user or the service provider could choose the authorisation service, as long as it is trusted by both parties.

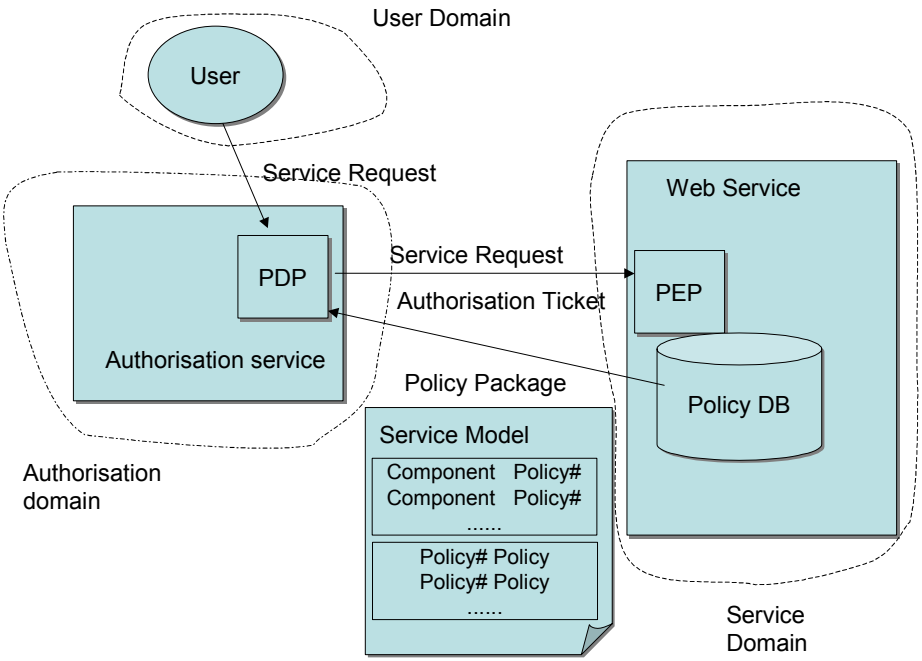

Figure 2: Policy, Service and User domains

In a traditional authorisation example, a third party can be used to enforce privacy between the two main parties. The authorisation domain must see the policies and the user's credentials but the credentials need not be passed onto the service provider. The use of a separate authorisation domain also means that the computation can be moved from the service infrastructure; the next section discusses how this can be achieved whilst avoiding bottlenecks associated with a central service.

Section 3.1 described some of the authorisation problems associated with composed web services; it is in these situations where there is a considerable advantage in creating a separate authorisation domain. Although Figure 2 shows a single web service interacting with a user, several services could have been added with all the services communicating authorisation information via this authorisation domain. This authorisation domain would be responsible for making policy decisions based on each services authorisation policies and the user's credentials. As with the traditional authorisation service, it is essential that any interested party including the users and the co-ordinating services cannot influence it. Additionally each party may want to keep their authorisation policies and credentials private from others in the transaction again supporting the need to carry out authorisation within a separate domain of control. All these factors mean that the authorisation domain should be thought of as a third party trust service.

4 Secure Hardware Authorisation Services

A third party authorisation service would typically be run as a central service and as such, it merely moves some of the computational bottlenecks to an additional web service. In doing so, it creates additional latency associated with the network traffic. Considerable risk is placed in running this central authorisation service – it becomes a point of attack for both getting bad transactions authorised and for observing the content of credentials and transactions. As such, a central service does not meet the basic goals for the proposed authorisation model of

providing an efficient authorisation framework that also preserves the privacy of the data involved in the authorisation process.

The basic algorithm run by the authorisation service requires looking at the service request; collecting the service policies; getting and validating credentials from the users and matching these with the policy requirements. Other functionality described in section 5 to deal with the multi-party composed services is achieved by extending this basic algorithm. As such, the authorisation service needs to be able to run the basic algorithm and have some form of network access.

The third party responsible for the authorisation service does not interact with the running process but it is responsible for the typical system management tasks of ensuring the system is running, available and secure. A *hardware security appliance (*HSA) [12] [13] offers an alternative mechanism for delivering such secure authorisation services by encapsulating a running service within secure hardware. Using this approach to service provision, an authorisation services can be placed anywhere from the clients to various services sites, with the secure hardware ensuring that the services cannot be subverted. Therefore, the authorisation decisions are trusted, privacy requirements are fulfilled and computation can be moved to the most appropriate position. Section 4.1 describes the HSA approach to service provision. Section 4.2 describes how this is applied to the authorisation service.

4.1 HSA Approach

The *hardware security appliance* (HSA) approach offers a mechanism for *logically* running a service within its own trust domain but that is *physically* located at the heart of conventional IT systems. The HSA itself is a tamper resistant hardware device; for example, based on a *hardware security module* (HSM), which provides a safe environment for services to run. Such a device has active hardware protection [14], which on detecting tampering will destroy cryptographic material – this protection ensures that the correct code is running and that any secret required by the service is highly protected and will not leak.

An HSM traditionally offers a cryptographic API such as PKCS#11 [15]; an HSA is a very similar – if not identical—physical device but with very different firmware allowing a service to be loaded, certified and configured. In doing so, the service binds together various critical security functions such as authentication, authorisation and audit along with cryptographic key usage into a simple service API. For example, an HSA based service could bind the authentication and authorisation processes with the use of a decryption key thereby securing access to decrypted files [12].

On a service being loaded into the HSA (Figure 3), it is configured with its own management policies and given its own identity (e.g. a PKI based identity where the service provider issues a certificate for the service). As well as the service, offering its normal functional API it also defines how it can be managed and the initialisation binds it very strongly to a service controller. The service now operates within its own trust domain – physically enforced by the tamper resistance of the secure hardware and logically enforced through the limited service API and these initial management policies. The management policies define not only who controls the service but the extent of their control (even specifying no management control). These management functions can be carried out remotely using the PKI identities of the HSA based service and the service controller.

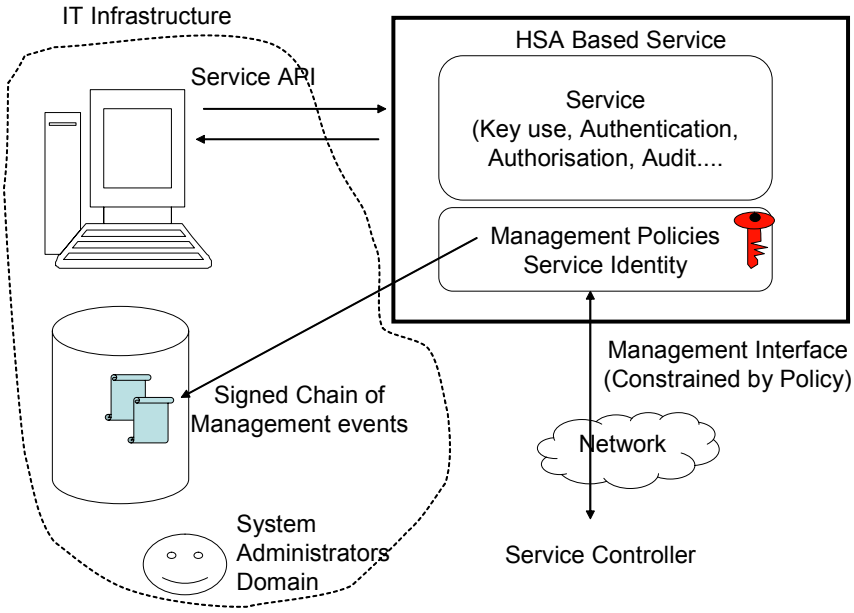

Figure 3: An HSA approach to service delivery

In effect, this changes the secure hardware device from one offering a simple cryptographic interface to a service delivery model. Simple cryptographic APIs [15] help protect keys from disclosure but a subverted process can lead to the misuse of keys. Research has shown [16] [17] [18] how further security can be achieved by putting more of the application within the secure hardware. The HSA approach takes this further by using the secure hardware as a service delivery device creating a strongly controlled domain for a limited but security critical process to run.

This approach can be contrasted with the TCPA [19] [20] approach that uses a simple and cheap trusted computing module to provide identity and measurement roots of trust. These roots of trust form an essential building block for gaining trust in an OS and therefore trust in the applications running on a computer system. Instead of providing a root of trust on which other layers can build, the HSA approach is to secure small but critical slice of the computing infrastructure by pulling it out as a service. Trust in this slice or service is then rooted in the secure hardware with the hardware security protecting the integrity and identity of the service.

4.2 HSA based Authorisation

This approach opens up the possibility of having an authorisation service provider delivering HSA based authorisation services over which they retain (limited) control. The third party will load service code, initialise and certify the service running within the HSA. It can now be shipped to those wishing to use the service.

The authorisation service itself is the same as if the third party was running it – it is the same basic code. The difference is now that the service provider no longer has a central server that is a bottleneck. They distribute their processing power by distributing the HSA to the most appropriate location. Instead of all authorisations being performed at the service side it can now be moved to the clients' sites. The bottleneck associated with policy computation (and networking for policy validation) is thus dissipated from the centre to the edges of the transaction.

The characteristics of the third party authorisation service have now changed considerably. Rather than operating and securing a set of servers seeing all transactions they now act as a certification authority. They load their service code into HSA along with policies defining

limitations on the service. This creates an instance of their authorisation service. This service instance then receives an identity by creating an RSA key-pair, which the authorisation service provider certifies (via an X509 Certificate) as a valid service that they control. The active protection provided by the hardware device will prevent this identity from being stolen – attempts to gain the key will lead to its destruction and therefore the revocation of the service.

This certification process allows the various parties to see why they should trust the authorisation service. A known service provider, on known secure hardware, and even using known code, provides the authorisation function. Each party can decide whether to trust the authorisation service based on this certificate. Wherever the HSA based authorisation service is located, the relying parties know that it is protected by the secure hardware and therefore cannot be subverted by any interested parties. In this way, a separate authorisation domain has been created.

Information such as authorisation policies and credentials can now be shipped directly to the HSA based service with appropriate encryption. They are only unencrypted within the protected boundary and therefore never become visible to those hosting the service or the service provider itself.

5 Distributed Authorisation Services

The HSA provides a mechanism for delivering an authorisation service and in doing so it brings some unique computational and privacy properties to a distributed system. This section examines the service that is provided within the HSA in more detail. Firstly, the traditional client/server authorisation problem is addressed – this describes the core mechanisms. This is followed by sections that describe how this HSA based authorisation service can be extended to create an authorisation domain for a multi party transaction.

5.1 Simple Authorisation

Figure 4 extends Figure 2 showing the role of the HSA based authorisation service within a simple client/server authorisation example. Here there are the two basic parties involved in the transaction as well as the authorisation service provider, credential providers and the authorisation service running within the client's IT infrastructure.

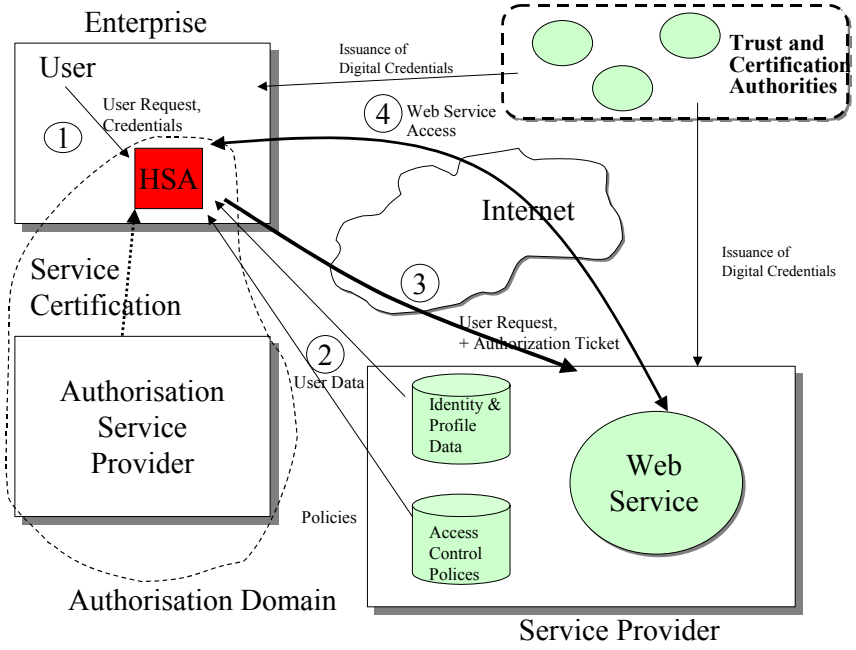

Figure 4: HSA based Authorisation

Traditionally the user's request with accompanying certificates and credentials would be sent straight to the service provider for verification. Instead, the request (1) and accompanying credentials are passed through the HSA based authorisation service.

This authorisation service must then obtain the authorisation policies and any additional authorisation data (2) from the service provider. The HSA will identify itself to the service provider (through its certificate ID) and request the current set of policies. The service provider can make a decision as to whether they trust the HSA based service and assuming they do they will return the policies. In cases where there is authorisation service or it is not acceptable to the web service provider, the transaction could proceed with the service provider running their authorisation rules.

The communication between the two parties could be done using SSL or TLS sessions which would ensure that policies received at the HSA have integrity and will not have been observed (even by the user's host system that may run the TCP stack). Alternatively, policy data could be transmitted as signed encrypted SMIME or XML structures. The secure session and policy exchange could prove expensive but both can be optimised where the two parties' interact regularly. Firstly, TLS session state can be stored and re-established reducing the full handshake. Secondly and more significantly, the HSA can cache policies (perhaps in an encrypted form on the host systems disks) which will only change occasionally. An exchange could firstly consist of a hash of the policies with the secure session only being established where a (confidential) policy exchange is necessary.

The HSA authorisation service now checks the web services policies against the user's credentials. In doing so, it may check credential revocation but CRLs may be cached between various service invocations; performing authorisation at the client site means that the client can download and maintain periodic CRLs. The authorisation service can now make a decision about the policy and it will generate a signed authorisation ticket that is then used to communicate its decision.

This authorisation ticket will be a signed document containing the *yes*, *no* decision along with the details of the request (e.g. the hash of the SOAP request) and details of the policy package used (e.g. the policy package version, hash and name). It may also include details of the credentials used or where disclosure is more sensitive it could include a hash of each credential along with say the public key of the user (for linking to secure communications with that user). The authorisation ticket is now an item that the web server can validate and it contains references to all information used in authorisation and is therefore a useful audit record. The web service must make a decision to trust the authorisation service (hence the ticket); this is probably done early in the communication session but should be confirmed on validating the tickets signature.

This ticket is now simply forwarded (3) to the web service provider along with the request thereby allowing the user to gain access to the services (where appropriate). The service provider must interpret the ticket but this is now a case of making a decision as to whether they trust the authorisation service and checking the signature. They can now allow the transaction with the user (4) or, where appropriate, block it. As described earlier, it is not necessary for the trusted third party to perform their computation on their site. This activity can be distributed to the client's computer or within the client's local IT infrastructure using an HSA to deliver an encapsulated distributed authorisation service. This now removes many of the bottlenecks associated with an authorisation server by distributing its computing power to the client side by issuing simple to manage hardware devices.

Having multiple parties generating and evaluating policies suggests standardisation of various parts of the authorisation process would smooth interoperability. The initial request (1) needs to be in a standard form as do the credentials – for example as a SOAP request accompanies by X.509 identity and attribute certificates. The authorisation service needs to be able to obtain a set of policies (2) and supporting service description [21] – this may be a simple web request to obtain the data but there is a need for a standard policy language. The authorisation

service issues a ticket for the service provider (3) which again must take a standard form (e.g. as it could be a SAML assertion [22]). This determines whether the client is authorised to access the web service (4) for the specified transaction.

A simple authorisation example has been described showing how an HSA based authorisation can be used. The authorisation service is provided within the secure hardware and credentials and policies need only be shared between the secure hardware and the owner of the data – hence preserving privacy of the data. Computation has also been moved to the edge of the transaction reducing the policy evaluation bottleneck at the web service.

5.2 Symmetric Authorisation

As web services offer complex services and as they are provided on a more ad-hoc basis, both parties will want to perform some form of authorisation. In the example scenario in section 2 a user booking their travel will want to check that a travel agent meets their (or corporate) policies as well as the travel agent needing to check the users abilities (e.g. to travel and to pay).

Two-way authorisation ensures that for a given type of task the user can specify a set of policies stating required aspects of a service (e.g. visa accredited). As well as the local HSA service checking the user's credentials match the web service's policies, the user can send their policies to an HSA based service run at the web services site. Here the user would expect to see an appropriate validation ticket. Where clients use standard policies, the service provider's authorisation service can cache tickets for the period of any credential revocation.

Alternatively, the client's HSA based authorisation service could check the policies getting credentials from the service provider. This approach fits nicely with the movement of computation to the edge of the transaction. Privacy is retained in that credentials will be passed through to the HSA based service and only unencrypted in the secure hardware.

This symmetric authorisation example relies on having the underlying authorisation domain that is protected from all interested parties. The computation is performed in secure hardware and which can be located either at both sites or just within the client's site.

5.3 Authorisation for Pre-composed Services

A well-planned composed web service would consist of a number of potentially independent web services each having their own set of policies that must be satisfied for any particular request. This could lead to a composed policy consisting of a conjunction of each of the service's individual policy (where there is a choice of services, a disjunction of different policies could be included). A service invocation must satisfy the overall composed policy in order to proceed successfully.

The combined policy will be passed to the HSA based authorisation service associated with the client that evaluates the policy producing a composite ticket. A composite policy could include a number of individual policies and the associated ticket would include a table of the hash of each individual policy along with the decision. This decision table is part of the signed authorisation structure and a constituent service getting a ticket can find the digest of their policies and check the decision.

For example, the travel service scenario described in section 2 would involve the travel service pulling together a number of policies for various services they offer such as airlines, car hire, etc. On authorising a travel package, the authorisation service checks the composite policy and each constituent service can check their policies have been satisfied by looking inside the authorisation ticket. They must of course check that they trust the authorisation service signing the authorisation ticket.

In the above discussion, the central service pulls together a policy set from the constituent services that will then be transmitted securely to the client's authorisation service. The constituent services will have no view on what policies other services are using but this

central service will. Policies could be requested from each constituent service for each transaction such that they would be encrypted for the user's HSA based authorisation service. This would loose some of the efficiency gains from pre-composing policies.

The symmetric authorisation section above suggested that the service side could also have an HSA based authorisation service so that the authorisation domain is extended from the edge of the transaction to the middle. For this type of example, this authorisation service would manage the policy set, combining policies provided by other constituent services. Privacy can now be preserved in that each service supplies its own policies to the authorisation domain. The full authorisation domain, for any transaction, is then formed by the co-operating set of HSA based authorisation services.

5.4 Authorisation for Dynamic Composition

The above discussion suggests a simple way to extend authorisation for composite services where there is a well-defined combination of services and associated policies. Often the situation will be more complex as a transaction is formulated in a more dynamic manner. Services can be added as required and may be derived from a dynamic service registry or through an auction. The user may want each constituent service to meet appropriate policies. In these cases authorisation becomes more of a dialog. For example, instead of having a simple book travel request it is more likely that the client interacts with a travel agent who in turn interacts with various service providers.

In these situations, there is an interaction as the service provider pulls together a transaction over multiple parties. To perform the transaction involving all these parties the client must satisfy the union of all the individual policies. Once the service provider knows each subpart of the transaction is satisfied, it can allow the client to commit to the overall transaction.

The service provider forming a composite policy from their set of partners and testing the client's ability to satisfy the transaction (as described in section 5.3) could do this. However, a failure to meet an aspect of this transaction could lead to the need to renegotiate how the transaction is provided.

Instead, the formation of a transaction could be accompanied by an authorisation session. As the co-ordinating service (e.g. the travel agent) finds a possible provider for an aspect of the transaction, they enter that services policies into this authorisation session and get a suitable ticket. A service provider may be chosen purely on the basis that the client has authorisation to use them but part of the authorisation session may be checking that the client's policies are also met. For example, a travel agent proposing a particular airline may send the airlines policies to the client so that they show they have the appropriate passport, visas and they would get the client to check that the airline meets any specific airline policies.

Once they have the set of appropriate tickets, they can let the transaction complete. The set of tickets with the common session ID then form the authorisation for the overall transactions – although individual services will only see the tickets relevant to them. The co-ordinating service needs not see each individual services policy or how it has been satisfied by the client's credentials; this knowledge remains only in the authorisation domain.

The authorisation session must be independent of all the parties involved in the transaction – each may have some reason for influencing the authorisation service; or trying to gain a view on policies and credentials involved in a transaction. The HSA based authorisation service provides a way of getting such an authorisation session and since the service has a (PKI based) identity, each party can talk securely to such a service. As with previous examples, this authorisation service can be placed on the client's site relying on the secure hardware to protect the integrity of the service. This results in the transaction being processed at the edge of the system.

More generally, a set of authorisation services running at various sites involved in the transaction can be combined to all work together in a single authorisation session (see Figure

5). As a party is introduced into the authorisation session, they may also introduce an HSA based authorisation service, which must be certified by a provider trusted by the other parties and the other authorisation services. The policy decision can now be computed by an HSA hosted by any party involved in the transaction.

As well as providing the policy decision engine, the authorisation services can help manage the users' or services' credential wallets and policy database. A party having such a service can then make all requests via their HSA-based authorisation service. This service may then manage disclosure of policies; for example, ensuring that they are encrypted and only sent to other trusted authorisation devices. It can also manage disclosure of credentials; as with policies ensuring that they are only disclosed to other authorisation services or even making sure that they execute policies referring to certain credentials.

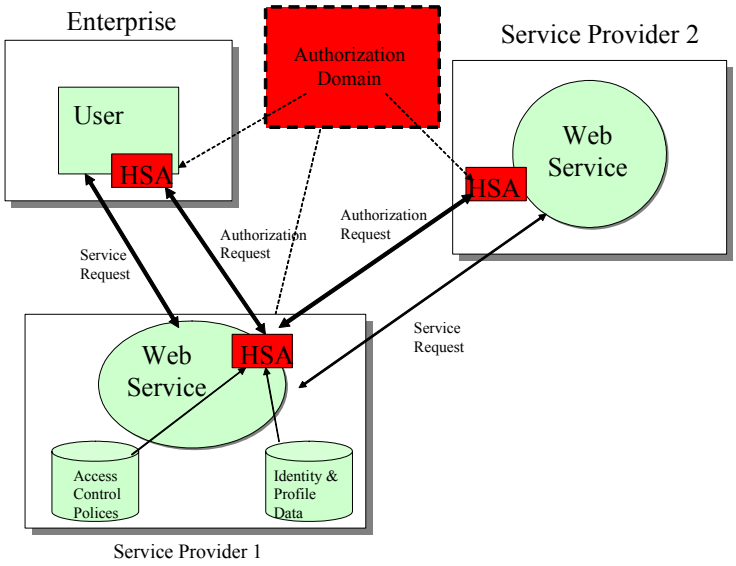

Figure 5: Multi-device authorisation sessions

6 Discussion

The provision of secure, trustworthy and efficient authorisation systems is a major problem when dealing with access control for web services in distributed and heterogeneous environments, such as B2B contexts, supply-chains and federated e-commerce sites. Classic centralised authorisation services or traditional distributed authorisation systems show their limitations.

Authorisation is typically thought of as a request, access control decision, followed by access to a service or resource. The model promoted here firstly allows form more complex access control policies that can be based on a variety of credentials, that can be matched with the service request in a manner specified by the policy. This is enabled due to the ability to move what could be complex computation towards the client (user) - thereby removing traditional bottlenecks. The third party nature of the service also makes strong privacy guarantees that mitigate some of the concerns often associated with using credentials in authorisation.

The proposed authorisation model also brings considerable flexibility allowing an authorisation session to be established underneath a service session. This enables complex service interactions, in that ad-hoc authorisation is now possible but the privacy and computational properties of the model are preserved.

A major issue with this type of solution is the cost associated with placing the trusted hardware on the client's side. Such solutions are not appropriate form typical consumer

transactions. The approach is appropriate where companies are regularly interacting with services maybe as part of supply-chain systems or corporate B2B portals.

7 Conclusion

This paper proposes a model combining a third party authorisation service approach with a hardware security service delivery model to achieve mobility of the authorisation policy's decision point. In a simple authorisation case, this has two major advantages: firstly, computation is moved from a single service bottleneck to the client's site thereby achieving a more even spread of workload. Secondly, having policies and credentials validated at the client's site brings significant privacy properties.

Web service frameworks offer possibilities of composed multi-party services that have more complex and dynamic authorisation requirements. The proposed authorisation model allows a number of HSA-based authorisation services to be formed into an independent authorisation domain enabling the free flow of authorisation information needed for such compositions.

8 References

[1] Richter, J., Applied Microsoft .Net Framework Programming. Microsoft Press (2002)

[2] Apte, N, Mehta, T, Web Services: a java developer's guide using e-speak Prentice Hall (2001)

[3] Karjoth G., The Authorization Server of Tivoli Policy Director, 17th Annual Computer Security Applications Conference, ACSAC 2001, December 10-14, 2001

[4] Baldwin A., Beres Y., Casassa Mont M., and Shiu S., Trust Services: A trust infrastructure for e-commerce. HP Labs TR HPL-2001-198 http://www.hpl.hp.com/techreports/2001/HPL-2001-198.html, 2001

[5] Housley R., Ford W., Polk W., Solo D., RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL profile, IETF, 1999

[6] Farrell S., Housley R., An Internet Attribute Certificate Profile for Authorization – IETF, 1999

[7] Rosenberry, W., Kenny, D., Fisher, G., Understanding DCE. O'Reilley & Associates, Inc., 1992

[8] Woo, T.Y.C, Lam, S.S., Authorization in Distributed Systems: a formal approach. In proceedings of the 13th IEEE Symposium on Research in Security and Privacy, pages 33-50, Oakland, California, May 4-6 1992

[9] Newman, B.C., Proxy-based Authorization and accounting for distributed systems. In proceedings of the 13th International Conference on Distributed Computing Systems, Pittsburgh, Pennsylvania, May 1993

[10] Woo, T.Y.C, Lam, S.S., Designing a Distributed Authorization Service, University of Texas at Austin, 1998

[11] Ryutov, T., Newman, C., Representation and Evaluation Policies for Distributed System Services, University of Southern California, 2000

[12] Baldwin, A., Shiu, S., Encryption and Key Management in a SAN, In Proceedings First IEEE International Security in Storage Workshop, Maryland, December 2002

[13] Baldwin, A., Shiu, S. Hardware Security Appliances for Trust, In Proceedings First International Conference on Trust Management. Springer Verlag, 2003

[14] Fips, Security Requirements for cryptographic modules. Fips 140-2 2001 http://csrc.ncsl.nist.gov/publications/fips/fips140-1/fips1402.pdf, 2001

[15] RSA Labs, PKCS#11 v2.11 Cryptographic Token Interface Standard, ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v211/pkcs-11v2-11r1.pdf, 2001

[16] Smith, S.W., Palmer E.R, Weingart S., Using a High Performance Programmable Secure Coprocessor. In Proceedings of the second international conference on financial cryptography. Lecture Notes in Computer Science. Springer-Verlag, 1998.

[17] Itoi N., Secure Coprocessor Integration with Kerberos V5.USENIX Security Symposium, www.citi.umich.edu/techreports/reports/citi-tr-00-2.ps.gz, 2000

[18] Smith S.W., D. Safford, Practical Private information retrieval with secure coprocessors. IBM Research T.J. Watson Research Centre, 2000

[19] Pearson, S. (ed.), Trusted Computing Platforms, Prentice Hall, 2002.

[20] Trusted Computing Platform Alliance, TCPA Main Specification, Version 1.1, http://www.trustedcomputing.org, 2001

[21] W3C, Web Service Description Language (WSDL) version 1.2 - http://www.w3.org/TR/wsdl12, 2003

[22] OASIS, SAML 1.0 Specification Set - http://www.oasis-open.org/committees/security/#documents, 2002