

A Privacy Preserving Enhanced Trust Building Mechanism for Web Services

Zhengping Wu, Alfred C. Weaver

Department of Computer Science, University of Virginia
151 Engineer's Way, P.O. Box 400740, Charlottesville, Virginia 22904-4740
{zw4j, acw}@cs.virginia.edu

Abstract

With the development and deployment of web services, information technology can more easily provide business services for clients. However, the lack of effective trust building mechanisms impedes the deployment of diverse trust models for web services. One important issue is the lack of mechanisms that can dynamically build trust relationships with privacy preserving. Current web service technologies encourage a client to reveal all the private attributes in a pre-packaged digital credential to the service provider fulfilling the requirements for verification. This may lead to a privacy leakage. We propose a mechanism whereby the client discovers the service provider's requirements from a web service policy document, then formulates a trust primitive by associating a set of client attributes in a pre-packaged credential with a semantic name, signed with the client's digital signature, to negotiate a trust relationship. Client privacy is preserved because only those attributes required to build the trust relationship are revealed. After negotiation, we propose a trust group element with dynamic validation to represent as well as to keep track of this built trust relationship.

Keywords: Trust building, Privacy preserving, Web services

1. Introduction

Trust is critical in business services. Web services have become an important media for online business services. Business services typically have large and dynamic client populations. The client of a web service and the service provider often share no prior relationship and no common security domain. While some web services permit anonymous access to low-sensitivity information such as weather reports, most require strong user authentication before allowing access to sensitive information such as electronic medical records protected by HIPAA [1]. This paper introduces a mechanism for conveying the minimal required set of client attributes in a pre-packaged credential to the service provider in order to create a trust relationship while simultaneously respecting client privacy and keeping track of the changes within this trust relationship.

Daily life provides many examples of the need for trust relationships to be built before any privilege is granted. For instance, governments insist upon proof of citizenship before issuing a passport; banks must verify a person's identity before opening an account; university libraries need to verify student status before lending books. In these examples:

- A client first initiates a request (issue a passport, open an account, check out a book);
- The service provider responds with a request for specific private attributes required by that organization's policies (birth certificate, driver's license, student ID);
- The client either provides the attributes required, in which case the request is usually granted, or the client fails to do so, and the request is denied.

Web services operate in an analogous way. If a student (client) attempts to open a free student checking account using an online web service, the service provider must verify the client's eligibility and the client must supply an acceptable security token that confirms the client's student status. A security token is a collection of security-related information that conveyed within a SOAP message [2]. Only then will the service provider allow building a trust relationship with the student and granting the access privileges for the student to open a new account.

Whether the credentials exchanged are paper documents, as in the first three examples, or security tokens, as in the web service example, the exchange of credentials can lead to the problem of information leakage. First, a credential may not be used for its intended primary purpose. For example, a driver's license is often used as a proof of age when its intended function is to document permission to operate an automobile. Second, a pre-packaged credential may reveal more information than is necessary. For example, a student ID may reveal the holder's degree information, an attribute not required by the bank to establish student status.

Another common way of building a trust relationship within a web service environment is for the service provider to require pre-enrollment of its clients before granting any access privileges. Here, a client could be either an individual user or a trust domain (a collection of users with common attributes). As discussed in Ferraiolo et al.'s papers [3, 4], pre-enrollment and conditioning of all future access upon the satisfactory presentation of

tokens that contain role information is required for all clients. However, this approach is contrary to the basic philosophy of web services, which anticipates choosing services dynamically at run-time.

The authorization of an access request for web services can also be based on a trust relationship built between the client and the service provider via credential exchanges. A pre-packaged credential such as a student ID token may be necessary to open a free checking account, but precisely because it is pre-packaged and standardized it may contain information (address, major, graduation date) that is superfluous to the single claim of student status that the bank actually verifies before opening an account.

In this paper we propose a mechanism that reveals the minimal number of attributes necessary to build the desired trust relationship for the web service environment. Using this mechanism, a set of attributes signed by the client's digital signature is associated with a trust primitive element. This element is used to negotiate a trust relationship. Any changes of policy requirements associated with this trust relationship are dynamically enforced using a trust group element. The remainder of this paper is organized as follows. Section 2, 3, 4 discusses the framework of our privacy preserving enhanced trust building mechanism. Section 5 describes the implementation information. Section 6 makes a comparative analysis and discussion. Section 7 gives a belief summary of related work. The final section summarizes our work's contributions.

2. Trust Building Related Facilities

Web services use the Simple Object Access Protocol (SOAP) [5] to exchange information. SOAP is a lightweight method for exchanging structured information in a decentralized environment. EXtensible Markup Language (XML) is used in SOAP to define an extensible messaging framework that can exchange a message over a variety of underlying protocols. Although SOAP is the basic infrastructure for information exchange between web services, it does not provide any security mechanisms for the information exchanged. Web service security is based on a process in which a web service requires that an incoming access request prove a set of claims such as name, public key, permission, or capability. A web service indicates its required claims and other security related attributes in its policy document (e.g., a WS-Policy [6] file). If an access request arrives without having the required proof of claims, the service provider ignores or rejects the request. Below are some other extant concepts or facilities for web service security.

- Security token: A security token is a collection of security-related information conveyed within the format of a SOAP message [2].

- Signed security token: A signed security token contains a set of related claims (assertions) that are cryptographically endorsed by the issuer.
- Security token service: A security token service (STS) is a web service that issues security tokens [2, 7]. A STS makes assertions based on evidence that it trusts to whoever trusts it. So part of the STS's functionality is a certificate authority for web service environments. To communicate trust, a STS also requests proof, such as a security token or a set of security tokens, and issues new security tokens with its own trust statements (note that for some security token formats, which embed other digital certificates such as X.509 or Kerberos tickets, this can be just a re-issuance or co-signature).
- Attribute service: An attribute service is a web service that maintains private attribute information about principals within a trust domain.

More recently, WS-Trust specification [7] enables applications to construct trusted message exchanges and provides a flexible set of mechanisms that can be used to support a range of security protocols. As described in WS-Trust specification, trust establishment is the mechanism by which one entity relies upon a second entity to execute a set of actions or to make a set of assertions about a set of subjects or scopes. But this specification does not mention how to build trust relationships and how to represent these trust relationships. Obviously, the trust relationships conveyed in these actions or assertions can be built through exchanging or brokering security tokens at run time. So the token-based mechanism provides an applicable solution to build trust relationships in a dynamic environment of web services.

3. Trust Primitive Element

Definition 1: A trust primitive is defined as the minimal subset of attributes in a pre-packaged digital credential, which has a complete semantic meaning according to a set of policy requirements. A trust primitive is signed by the credential holder, which is either an individual user or a security domain.

In the proposed privacy preserving enhanced trust building framework for web service environments, a trust primitive is represented as a subset of attributes in the attribute service and conveyed as an XML element when it is exchanged between security domains. As illustrated in figure 1, trust primitive 1 corresponds to an electronic library access rule with three required attributes (4, 6, and 7). The holder of the digital credential can form this trust primitive element and will be allowed to use the electronic library if the server can verify that its three attributes are valid (that is, if the token has been issued by an acceptable authority, the token is not expired, and the

holder operates in the role of student, faculty or staff). Since neither the name nor ID number is part of the set of attributes associated with this trust primitive, anonymous access is permitted. Trust primitive 2 is for library checkout. It contains the same three attributes in trust primitive 1 plus attribute 2, ID number, which is needed for library accounting. Trust primitive 3, consisting of attributes 3, 6 and 7, might be used to verify same-sex gender before granting entrance to a dorm floor, or by substituting attributes 1 or 2 for attribute 3 it might be used to verify a specific individual's residence on the dorm floor before granting entrance.

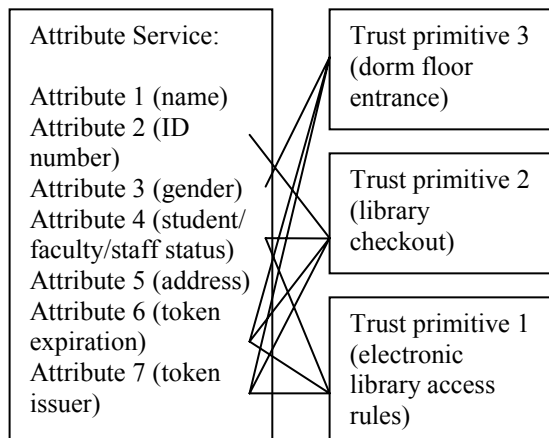


Figure 1. Attributes and three possible trust primitives

A credential holder forms a trust primitive voluntarily. Every request of retrieving a trust primitive from an outside domain will be verified by the attribute service to selectively disclose the subset of attributes associated with the trust primitive. Another merit of trust primitives is that they prevent initiation of selective disclosure from anyone except the credential holder. Figure 2 shows the workflow of the proposed protocol working with trust primitives for privacy preserving in trust building process.

In this protocol, a client and a service provider (two principals) from different security domains are assumed. Before the beginning of this privacy preserving enhanced trust building workflow, the attributes of the client are stored at the attribute service of the client's security domain. The client is also assumed to hold a security token containing its identity and other security related information. The workflow is initiated by a need for the client to disclose some of its attributes to the service provider for negotiation. This need could be triggered by the access policy of the service provider's security domain, which is publicly accessible via a policy document, or by the service provider's direct request for the client to provide one or more attributes. Then the workflow executes steps one through ten to finish a round of information exchange for trust building process. Step numbers below correspond to the numbered arrows in figure 2.

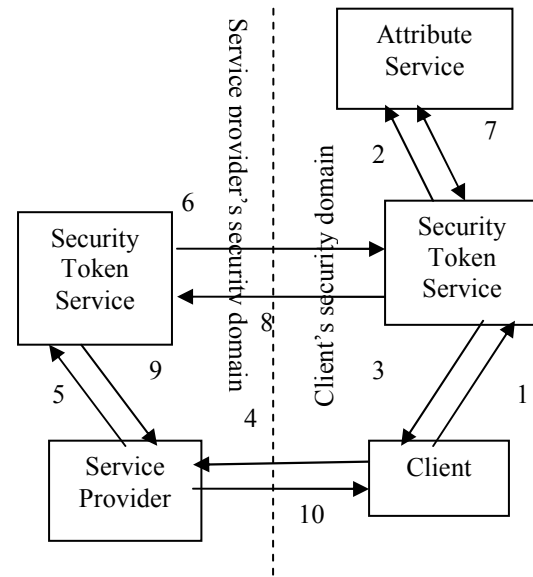


Figure 2. Workflow of Privacy Preserving Protocol for Trust Building Process

- 1) After reading the service provider's access policy or receiving a request for proving some attributes to obtain access, the client initiates a service request, which contains the trust primitive corresponding to the policy or request for attributes from the service provider, to the STS in its own security domain. The security token held by the client is also embedded in this service request message for proof of identity.
- 2) When the STS receives the service request, the STS extracts the security token from the message, verifies the legitimacy of the security token, and registers the trust primitive at the attribute service in its own security domain.
- 3) Then the STS adds this trust primitive to the client's security token, re-signs the security token, and sends it back to the client.
- 4) When the client gets the newly signed security token, the client embeds the security token in the access request and sends the request.
- 5) After receiving the client's access request, the service provider asks its own STS to check whether the request complies with the service's access policy.
- 6) The STS of the service provider sends a request for attribute verification to the STS of the client. The newly signed security token of the client is sent together with the request.
- 7) The STS of the client extracts the trust primitive, which corresponds to the access policy of the service provider, and uses this trust primitive as the query keyword to search the attributes disclosed by the client at its attribute service.
- 8) The STS of the client returns the attributes retrieved from the attribute service.

- 9) The STS of the service provider performs verification and sends its decision regarding the access request (granted/denied) to the service provider.
- 10) If access is granted, the service provider performs the requested operation and returns information to the client; otherwise the provider issues a denial.

Sometimes the client also needs to verify some of the service provider's attributes to negotiate a trust relationship, so the roles of the client and the service provider can be interchanged. Several rounds of exchange form the negotiation needed to build a trust relationship between the two principals.

4. Trust Group Element

Definition 2: A trust group represents a group of partners who comply with the same set of policy requirements. The partners here are entities who have direct trust relationships with the policy holder.

A trust group name is associated with a set of policy requirements. For example, if the service provider creates a set of policy requirements in the form of a WS-Policy file for negotiation of a trust relationship, the trust group name will be attached at the end of the file. Every partner complying with this set of policy requirements will use this trust group name to represent the corresponding trust relationship. A WS-Policy file containing a trust group looks like this.

```
<?xml version="1.0" encoding="utf-8" ?>
<policyDocument
xmlns="http://schemas.microsoft.com/wse/2003/06/Policy"
>
  <policies
xmlns:wssp="http://schemas.xmlsoap.org/ws/2002/12/sect"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
>
    <wsp:Policy wsu:Id="trustlevelsec-token">
      <wssp:SecurityToken wsp:Usage="wsp:Required">
        <wssp:TokenType>http://www.contoso.com/tokens/customXml#TrustLevelSecToken</wssp:TokenType>
        <wssp:TokenIssuer>http://www.cs.virginia.edu/TrustLevelSTS.ashx</wssp:TokenIssuer>
      </wssp:SecurityToken>
      <wssp:SecurityToken wsp:Usage="wsp:Required">
        </wssp:SecurityToken>
    </wsp:Policy>
```

```
</policies>
<trustGroup>TG001
</trustGroup>
</policyDocument>
```

After negotiation, a new trust group element is added to the client's security token, and the security token is signed by the client's STS again. Alternatively, the security token is replaced by a new security token issued by the service provider's STS, which contains the corresponding trust group element representing the new trust relationship. A trust group element is represented by three XML tags (trust group name and two domain/individual identities). Every policy holder also needs to record all the trust group elements that the holder is involved, and so do the partners. The trust group element looks like this.

```
<trustGroup>
<trustGroupName>TG001</trustGroupName>
<domain1>http://abc.com/localSTS.asmx</domain1>
<domain2>http://def.com/localSTS.asmx</domain2>
</trustGroup>
```

Access requests after the negotiation are granted by a verification of the trust group element in security tokens. If the access requirement for a trust group element changes in a service provider's policy, then the service provider needs to revalidate the previous trust relationship by invalidating the old trust group element, asking for the client's trust primitives again, and then verifying whether the new trust primitives meet the requirements of the changed policy. If access is granted, a new trust group element will replace the old one for future use.

5. Implementation

The architecture of our privacy preserving enhanced trust building mechanism is illustrated in figure 3. The web service client and web service provider build a dynamic trust relationship via negotiation engines and security token services in both security domains. The security token service in our implementation also includes a set of web services to interpret and exchange security tokens. At the same time, the security token service uses attribute service to register trust primitives for the client. Negotiation engines control the overall workflow to build trust relationships dynamically, which include the implementation of the protocol described in section 3.2 for a single round of negotiation.

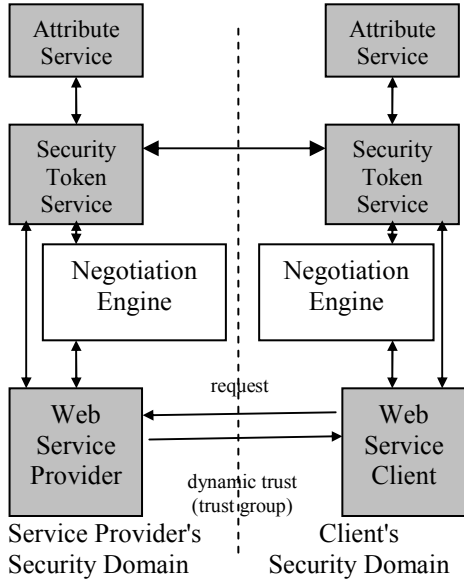


Figure 3. Diagram of the system architecture

We have implemented this system architecture using the Microsoft .Net platform. The .Net framework together with the Web Service Enhancement (WSE) 2.0 SDK, which supports the WS-Security [8] and WS-Trust standard [7], provides a complete platform for our system implementation. All the building blocks in our system architecture use web services as internal interfaces. We also have a graphic user interface to assist clients to define and sign their trust primitives, which is showed in figure 4.

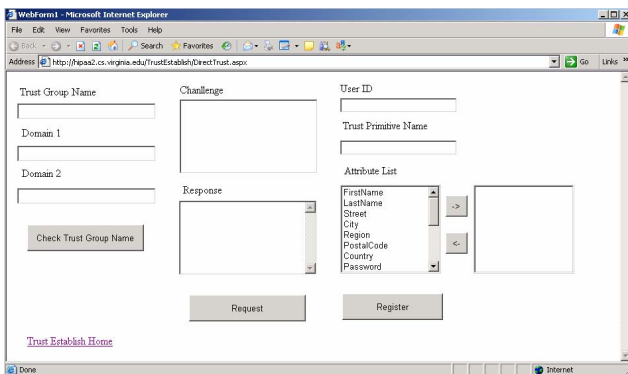


Figure 4. Graphic User Interface for Trust Primitive Definition

This privacy preserving enhanced trust building mechanism is a subsystem of a federated cyber trust system [9]. It provides the functionality of negotiating and building trust for a web service environment. When the federated cyber trust system is applied to a healthcare environment with hospital, pharmacy, insurance and billing security domains, the shaded boxes shown in figure 5 are the modules involved in the trust building

subsystem. Boxes with dashed border lines represent different security domains; boxes with solid border lines represent modules in the hospital domain; arrows represent information flows or interactions. All the modules use web services as interfaces for their interactions.

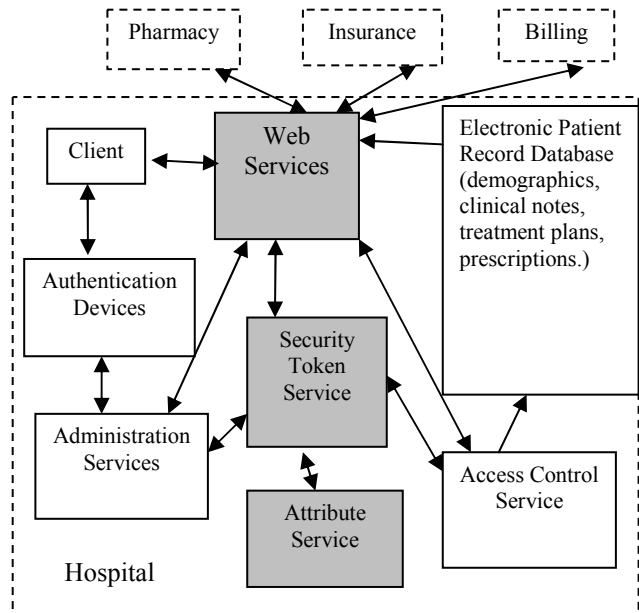


Figure 5. Federated cyber trust system for a healthcare application

The final thing needs to mention is token formats. Among all the interactions between modules, there are two different types of interactions. One is the interaction between modules within a security domain. The other is the interaction between security domains. To promote efficiency, we use user name tokens for intra-domain interactions, because they are more concise. To promote interoperability, we use SAML-formatted [10] security tokens for inter-domain interactions, because they follow industrial standards. Figure 6 gives examples of two token formats before embedding into SOAP message.

<pre><UsernameToken> <CreateAt>04/20/2005 8:00:00 am </CreateAt> <ExpireAt>04/20/2005 5:00:00 pm </ExpireAt> <UserID>123</UserID> <TokenIssuer>http://abc .com/TrustSTS.asmx</T okenIssuer> <TrustGroup>Name="T G001" Domain1="http://abc.co m" Domain2= "http://def.com"</Trust Group> <TrustPrimitive>TP001 </TrustPrimitive> </UsernameToken></pre>	<pre><saml:Assertion Version="2.0" ID ="ABC" IssueInstant="timestamp "> <saml:Issuer>http://abc. com/TrustSTS.asmx</sa ml:Issuer> <saml:Conditions NotBefore="04/20/2005 8:00:00 am" NotOnOrAfter="04/20/2 005 5:00:00 pm"> <saml:Subject><saml:N ameID>123</saml:Name ID></saml:Subject> <saml:Attribute>Name= "TrustGroup" NameFormat="urn:oasis :names:tc:SAML:2.0:att rname-format:basic" <samlAttributeValue>T G001</samlAttributeVa lue> Domain1=" http://abc.com" Domain2=" http://def.com"</saml:A ttribute> </saml:Assertion></pre>
UserName Token	SAML Token

Figure 6. Token Formats

6. Discussion

At this moment there are no standard criteria for evaluating trust building mechanisms, even no for trust management systems. Seamons et al. [11] proposes some requirements for policy languages for trust negotiation. Some of the requirements for expressiveness and semantics are also important in trust building, such as well-defined semantics, monotonicity, credential combinations, and transitive closure.

- Well-defined semantics: Our web service trust building mechanism has well-defined semantics, so that a user can easily and confidently choose any subset of all available attributes for disclosure. At the same time, the trust primitive and trust group elements are independent of any particular implementation.
- Monotonicity: Disclosure of additional attributes or credentials in our trust building mechanism guarantees no reduction of privileges, because additional attribute disclosure corresponds to satisfying a new policy or fulfilling an additional

requirement for additional privileges during a trust negotiation.

- Credential combinations: Our trust building mechanism provides selective disclosure of attributes, so that arbitrary credential combinations can be easily achieved.
- Transitive closure: Our trust building mechanism does not guarantee transitive closure, because the trust primitive and trust group elements are only valid for the participants involved in the trust building process directly. The extension of our mechanism to allow privacy control and protection during delegation may provide some sort of transitive closure capability in the future.

Comparisons with extant systems or specifications are also helpful to evaluate our privacy preserving enhanced trust building mechanism. In the joint security whitepaper from IBM and Microsoft [2], the WS-Trust specification describes the model for building trust relationships; the WS-Federation specification defines how to construct trust relationships using WS-Security, WS-Policy, WS-SecureConversation, and WS-Trust, and suggests ways for building trust relationships manually. To build a trust relationship using the WS-* specifications, the client sends the security token to the service provider; the service provider uses this security token to acquire corresponding attributes for verification of the client's compliance with the provider's access policy, and grants or denies access based upon the decision made by the verification process. In these WS-* specifications, there is no privacy protection in either the attribute service or the security token service which would allow the client to disclose attributes selectively. The client must always be willing to reveal all the attributes to the service provider to build a trust relationship. In contrast, our proposed trust primitive element provides a selective disclosure capability to protect the client's privacy.

To provide privacy preserving for trust building process in a web services environment, the client's security domain can limit the access capability of attributes to the client only. In that scenario, if the service provider needs to verify some attributes of the client, every attribute needed by the service provider has to go through the security token service and the client first. Compared to our proposed mechanism, these repeated procedures of transferring and signing attributes significantly reduce the efficiency of the trust building process. At the same time, our proposed mechanism also provides a privacy protection mechanism to prevent the revelation of attributes to an unauthorized entity either inside or outside the local security domain. For example, if a malicious hacker (or even a legitimate third party) gets a security token from the client's security domain, the hacker can pretend to be a user in the client's domain and ask for some attributes of the client. But in our proposed mechanism, trust primitives are initiated and

signed by the client only. Any unauthorized request for attributes will be invalidated by the absence of the client's digital signature, which is provided only by the client. Our proposed mechanism maintains operational efficiency while supporting selective disclosure and preventing unauthorized disclosure to hackers.

7. Related Work

Several types of trust building mechanisms have been described in the literature for service-oriented architectures. The most basic way is to map the identity of the client (or one identity in the client security domain) to one identity in the service provider security domain. Some other extant trust management systems build trust relationships between different security domains using the client's role as a basis for mapping. More recently, group-based mechanisms have been proposed to build trust relationships.

While "role" is an abstract concept, in a complex organizational setting such as a healthcare environment one might assign differing roles, and hence differing access permissions, to physicians, technicians, and patients. In Sandhu et al.'s paper on role-based access control [12], role-based trust building process is implied by setting a mapping between local roles and roles within remote domains, which we call a role-to-role mapping. In Chadwick et al.'s paper [13], the authors propose using X.509 certificates to manage trust relationships. This trust building process still employs a manual configuration of static role-to-role mappings to form a trust relationship before an actual access occurs. In the mechanism proposed by Freudenthal et al. [14], predefined trust relationships are used to complete the trust building process for dynamic cross-domain environments. The common approach of [12], [13], and [14] is that the authors try to decide whether to grant access at run-time by deciding what permissions each client has according to the assigned role and the predefined trust relationship associated with that role. The use of role as a basis for trust building creates a problem in domain-to-domain interactions, which is the potential misalignment of the precise definition of roles from one domain to another. One consequence could be that the domain whose users are requesting access might legitimately need to create special roles that map more precisely to the agreed intent of the requested operation. In that case, the users would be enabled by the policies of the domain they are accessing to take actions beyond those allowed them by their own domains' policies. A more general problem with roles is that their use generally does not conform to the principle of least privilege, as promoted by Schneider [15], which limits the security privileges to actual needs.

Vandenwauver et al. [16] and Van Dyke [17] both use group-based mechanisms to describe a collection of security requirements agreed to by the administrators for a

group of domains. Those group-based mechanisms reduce the administrator's burden of creating an explicit policy to manage each trust relationship. Both authors also assume that a recognized consortium of group members has created a trust group with predefined membership requirements. The service provider has to verify some non-identity attribute information about each client's service request. These trust group mechanisms lack the process of negotiation before building a trust relationship. Li et al. [18] proposes an entire trust management framework that can group logically related objects so that permissions about them can be assigned in one operation. These logical groups are defined by an authority, not via a negotiation process. All the group-based mechanisms mentioned above require some superior authority to create or predefine group information, and thus the resulting trust building mechanisms are not fully dynamic; they may not keep pace with the changing policies and requirements of the service providers.

The most important point is that all these identity-based, role-based and group-based mechanisms do not address privacy issues, which could lead to superfluous information disclosure during the trust building process. We propose a privacy preserving enhanced trust building mechanism that maintains the idea of negotiation but without inadvertent disclosure of unnecessary information.

8. Conclusion

In this paper we described a privacy preserving enhanced trust building mechanism that extends the extant trust building mechanisms for web services to gain many advantages from its privacy control and dynamic capabilities. Our research motivation comes from the complicated privacy requirements inherent to current healthcare data management and similar sensitive information management. Our new trust building mechanism is dynamic with these advantages:

- It allows only the client to choose what attributes may be viewed by the service provider. Therefore, it is capable of enforcing the client's privacy.
- It allows only the chosen attributes to be viewed by the service provider. Therefore, it is capable of disclosing private attributes selectively.
- It allows any trust relationships to be renewed whenever the service provider's policy is updated. Therefore, it is inherently dynamic.

Our future work will focus on the extension of trust primitive and trust group elements to allow privacy control and protection in indirect trust building and delegation.

9. References

- [1] Health Care Portability and Accountability Act, Public Law 104-191, <http://aspe.hhs.gov/admsimp/pl104191.htm>, August 1996.
- [2] A Joint White Paper from IBM Corporation and Microsoft Corporation, "Security in a Web Services World: A Proposed Architecture and Roadmap", <http://msdn.microsoft.com/library/en-us/dnwssecur/html/securitywhitepaper.asp>, April 2002.
- [3] D.F. Ferraiolo, J. Cugini, and D.R. Kuhn, "Role Based Access Control: Features and Motivations", Proceedings of 1995 Computer Security Applications Conference, December 1995, pp. 241-248.
- [4] D. Ferraiolo et al. "Proposed NIST Standard for Role-Based Access Control". ACM Trans. Information and System Security (TISSEC), August 2001, 4(3) pp. 224-274.
- [5] Anthony Nadalin et al., "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>, March 2004.
- [6] Siddharth Bajaj, et al., "Web Services Policy Framework (WS-Policy)", <http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-policy.asp>, September 2004.
- [7] Steve Anderson, et al., "Web Services Trust Language (WS-Trust)", <http://msdn.microsoft.com/ws/2005/05/ws-trust/>, February 2005.
- [8] Don Box, et al., "Simple Object Access Protocol (SOAP) 1.1", <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>, May 2000.
- [9] AC Weaver, SJ Dwyer, AM Snyder, J Van Dyke, J Hu, X Chen, T Mulholland, "Federated, Secure Trust Networks for Distributed Healthcare IT Services", Proceedings of IEEE International Conference on Industrial Informatics, August 2003, pp. 162-169.
- [10] Scott Cantor, et al., "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>, March 2005.
- [11] K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu, "Requirements for Policy Languages for Trust Negotiation", Proceedings of 3rd International Workshop on Policies for Distributed Systems and Networks, Monterey, California, June 2002.
- [12] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman, "Role-based access control models", IEEE Computer, 1996, 20(2), pp. 38-47.
- [13] David W. Chadwick, Alexander Otenko, Edward Ball. "Implementing Role Based Access Controls Using X.509 Attribute Certificates", IEEE Internet Computing, March-April 2003, pp. 62-69.
- [14] Eric Freudenthal, Tracy Pesin, Lawrence Port, Edward Keenan, Vijay Karamcheti, "dRBAC: Distributed Role-based Access Control for Dynamic Coalition Environments", Proceedings of 22nd International Conference on Distributed Computing Systems, 2002, pp. 411-420.
- [15] Fred B. Schneider, "Least Privilege and More", IEEE Security and Privacy, September-October 2003, 1(3), pp. 55-59.
- [16] M. Vandenwauver, R. Govaerts, J. Vandewalle, "Role based access control in distributed systems", Communications and Multimedia Security, 1997 volume 3, pp. 169-177.
- [17] James Van Dyke. "Establishing Federated Trust Networks among Web Services", B. S. thesis, University of Virginia, March 2004.
- [18] Ninghui Li, John C. Mitchell, William H. Winsborough. "Design of a Role-Based Trust-Management Framework", Proceedings of the 2002 IEEE Symposium on Security and Privacy, Washington, DC, USA, pp. 114-130.