Proceedings of the Third International Conference on Web Services
(ICWS'03), June 2003, pp. 235-242

A Stateless Network Architecture for Inter-enterprise Authentication, Authorization and Accounting

H.T. Kung, F. Zhu and M. Iansiti
Harvard University
Cambridge, MA, U.S.A.

Abstract *Providing network infrastructure for authentication, authorization and accounting (AAA) functionalities required by inter-enterprise business applications operating over the global Internet is a challenging problem. The infrastructure needs to support large numbers of clients and services, and also to provide secure resources sharing between applications and across organizations.*

This paper describes a scalable and secure network infrastructure architecture for inter-enterprise AAA services, called .TRUST. The architecture has two novel features: (1) it uses a stateless design for improved security and simplified system structures, and (2) it supports a resource-sharing infrastructure while allowing decentralized management. To illustrate the use of the .TRUST architecture, the paper considers three application examples for which laboratory prototypes have been implemented.

Keywords *business network infrastructure, authentication, authorization, accounting, stateless systems, web services*

1. Introduction

New Internet standards such as XML (eXtensible Markup Language), SOAP (Simple Object Access Protocol), UDDI (Universal Description, Discovery, and Integration) and WSDL (Web Services Description Language), have enabled organizations to interoperate and share resources across organizational boundaries. This means that there is a need to support authentication, authorization and accounting (AAA) functionalities between applications and across organizations. An authentication system is used to verify the user's identity. Once the system knows who the user is, an authorization system decides what the user can do. Finally, an accounting system collects information of the service usage for various purposes such as usage monitoring, trend analysis and billing.

We postulate that an ideal network infrastructure for inter-enterprise AAA operations will need to satisfy the following properties:

First, the AAA infrastructure must provide a secure way for organizations to share resources and provide services across organizations. Today we often experience inconvenience and inefficiency like this: an employee of company A visiting company B is not allowed to use the latter's service even when there is an existing partnership relationship between the two companies. The employee will need to use an alternative service provider, pay for the service himself and subsequently file an expense report at company A for reimbursement. An AAA infrastructure should provide functionalities to allow company B to provide the service to this employee, account for the usage, and settle the expense with company A later. In the meantime, the system should respect privacy of users and protect resources from unauthorized use.

Second, the AAA infrastructure must be resilient in defense against security attacks such as data-mining and denial-of-service (DoS) attacks. Data-mining refers to unauthorized gathering of information for improper use. The threat of data-mining increases when services and data are shared among organizations. DoS attacks are aimed at depleting server or network resources that are necessary to provide services. DoS attacks have become one of the most serious threats to network-resident services in recent years. It has been reported that nearly 4,000 DoS attacks are launched each week in the year 2001 [6, 8]. To avoid resources depletion, it is desirable that servers in the AAA infrastructure do not keep any session state information about ongoing interactions with clients. That is, it is desirable that all servers in the infrastructure are stateless.

Third, the AAA infrastructure must be convenient for users to use. For example, as the number of Web services continues to grow, managing login username and password for each subscription service has become a significant burden to many customers. It has been reported [13] that more than 10 percent of customer-service interactions for over one third of all Web service providers involve customers who have forgotten their passwords. Thirty eight percent of these service providers were de-emphasizing strategic investments in authentication systems due to the cost, complexity, and inflexibility of existing systems [13]. Thus, by alleviating customers' burden, the AAA infrastructure will also reduce operating costs for service providers.

Fourth, due to the heterogeneous nature of enterprise systems, the AAA infrastructure needs to be open to work across platforms, security models and applications. In other

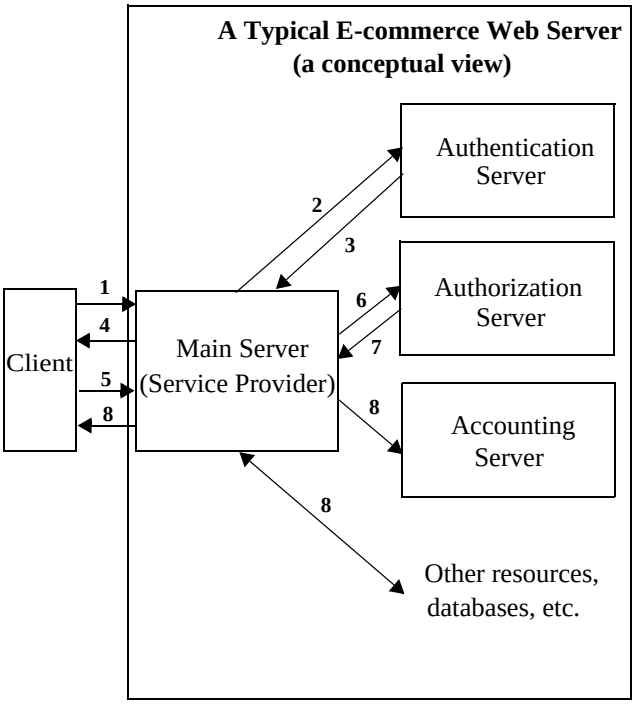


Figure 1. Traditional centralized AAA approach, where a central Web server keeps the states of all clients' transactions and implements all AAA functions. **1.** A client sends his ID and password to the main server. **2.** The main service forwards (ID, password) pair to the authentication server for verification. **3.** The authentication server notifies the main server of the result. **4.** The main server forwards the result to the client. **5.** If authenticated, the client sends service request to the main server. **6.** The main server sends the request and the client ID to the authorization server. **7.** The authorization server determines whether the client is permitted to perform the requested tasks and sends this information back. **8.** If the client is authorized, the main server provides it with the service requested. In the meantime, the accounting server keeps a record of service usage. The main server may acquire other resources for providing the service.

words, the AAA infrastructure should not mandate anyone who wants to interoperate to deploy identical systems.

Traditional AAA infrastructures, such as the PPP (point-to-point protocol) dialin system used by many ISPs [3] for providing internet access services, are unable to meet these objectives due to their centralized approach. Figure 1 illustrates a typical structure of this kind, which is adopted by most commercial Web systems today [3, 12]. Centralized approaches, in which a central server maintains clients' profiles, keeps the states of their transactions and implements all three AAA functions, are often specific to the services provided and lack of a trust-based mechanism that can work across organizations. Therefore, it is difficult to extend their usages to multiple service providers.

Currently, many companies are working on software products to support AAA functionalities across organizations [4, 9, 18]. Among these developments, the most notable ones are the Liberty infrastructure by the Liberty Alliance [9] led by Sun Microsystems and Microsoft's TrustBridge [18].

For the rest of this paper, we first describe he approaches used by Microsoft and the Liberty Alliance in Section 2. Then in Section 3, we introduce a stateless AAA network architecture, called .TRUST, for which we have implemented a laboratory prototype at Harvard. Section 4 discusses the design principles and properties of this architecture, and compares it to the classical Kerberos authentication system [20]. In Section 5 we illustrate these properties and their significance with three application examples which we have implemented on our .TRUST prototype. Section 6 concludes the paper.

2. The Liberty infrastructure and Trustbridge

Both the Liberty infrastructure and TrustBridge are designed to support cross-organizational resources sharing using "federated identity". Federated identity is the ability to recognize and leverage user identities among trusted organizations, without requiring users to re-enter their names and passwords [14]. In both infrastructures, identity federation is achieved by enlisting a trusted third party to help users complete the authentication process with the service providers.

We illustrate the overall Liberty architecture in Figure 2. Here, the identity provider serves as a trusted third party. It maintains identity information for each user, which may include information such as his home phone number, address, entertainment preferences, education history, etc. The identity provider and service providers form a circle of trust. There are two scenarios on how a user can get his identify to be federated at a number of Web sites. First, when a user logs in at the identity provider, the identity provider will notify the user of the possibility of federating

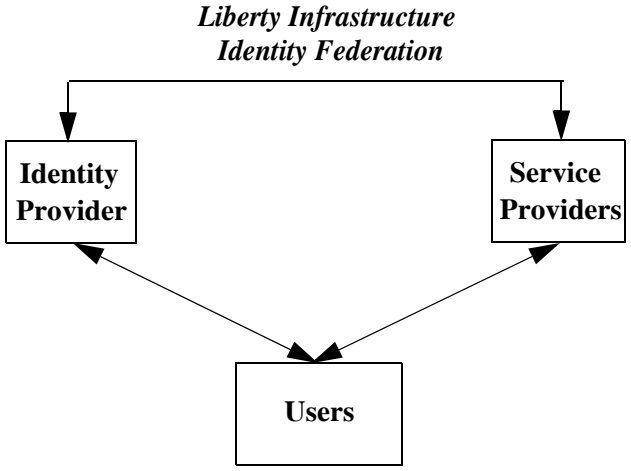


Figure 2. Overall Liberty architecture [9].

his local identity with service providers within the trusted circle and will solicit permission to facilitate such a federation. Second, when a user logs in at one of the Liberty-enabled Web sites, he will be asked whether to federate his local identity with his identity at the identity provider. In both cases, if the user chooses to federate his identity, his authentication state is reciprocally honored between these service providers and the identity provider. Subsequently, the user will be transparently logged into the Web sites with which he has established identity federation when he hits these sites. The identity provider remembers the user's login status at each service provider at which he has established identity federation. Therefore, when the user decides to log out from all members in the trusted circle, the identity provider may act as a proxy to communicate a logout request to each of these service providers.

With appropriate security schemes such as Kerberos and X.509 certificate, the Liberty infrastructure will allow employees from one organization to securely share resources with other organizations within the same trusted circle. Identity federation also enables employees from one organization to log in either at an identity provider or a service provider and prove their identity just once to gain access to a variety of resources. Furthermore, as the Liberty infrastructure is Web-based, it can interoperate with any platforms with standard Web functionalities and work with any Web-based applications. Therefore, the Liberty infrastructure satisfies our first, third and fourth property described in Section 1. The Liberty infrastructure fails to satisfy the second property because once a user logs into one of the Liberty-enabled servers, the identity provider and each of the service providers with which the user establishes identity federation will initiate a session state for that user. This means, as we discussed earlier, that any server in the Liberty infrastructure can be vulnerable to DoS attacks.

The Liberty infrastructure has some additional shortcomings. For example, it does not allow users to have full control over their private information. That is, users are not given the opportunity to decide the kind of information to share with individual service provider. Often service providers will receive more information than needed to complete service requests. For example, a service provider in the Liberty infrastructure can obtain a full set of information about a user from an identity provider even if it only needs to know his current employer to provide services accordingly. It is also possible that a user decides to federate his identity at a number of Web services but later chooses not to visit some of them. These Web services will still have to maintain session states for that user until these sessions expire or the user chooses to log out from the trusted circle. In such cases, network resources are wasted.

Microsoft's TrustBridge is also designed to achieve cross-organizational resources sharing through identity federation. It allows businesses that use Windows Active Directory to recognize and share identities with other organizations running Windows, .NET Passport service, or other Kerberos-based systems that support WS-Security protocol [18]. Figure 3 illustrates a typical scenario of how TrustBridge establishes cross-organizational resources sharing. Here Windows Active Directory serves as the identity provider for employees at each organization. TrustBridge, which runs across organizations, performs identity federation to authenticate employees of one organization to other organizations.

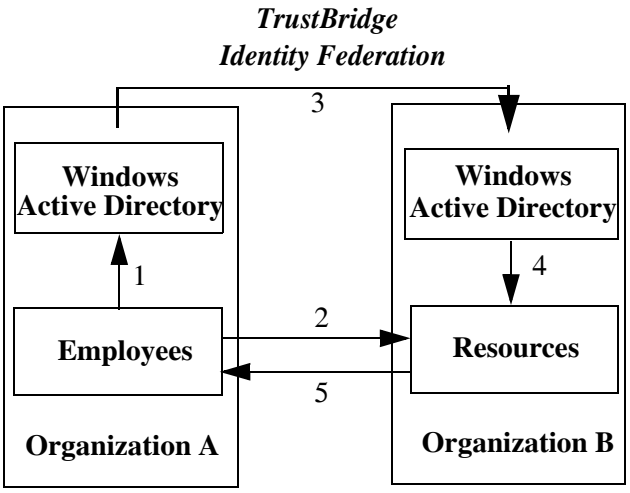
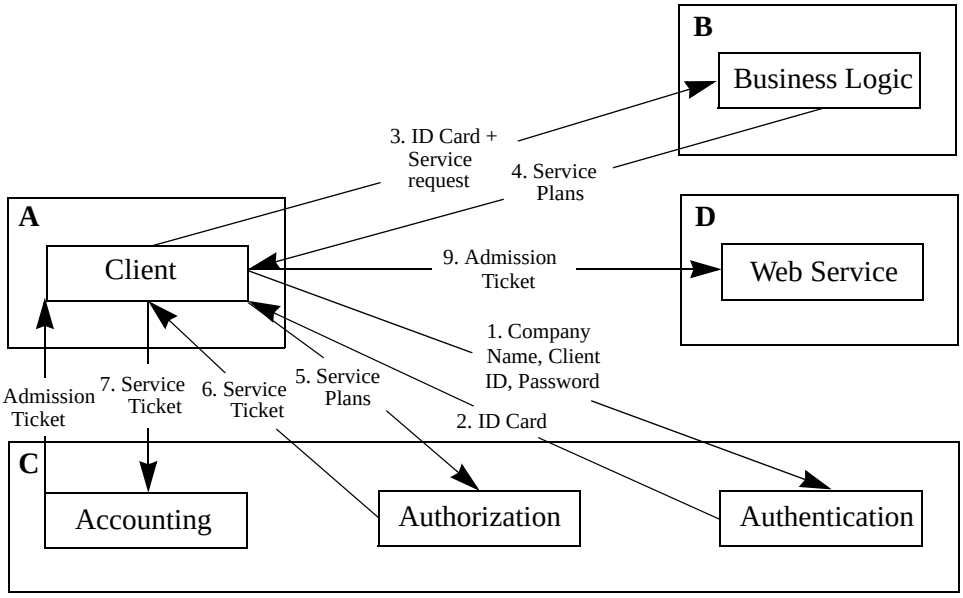


Figure 3. A typical scenario of using TrustBridge to achieve cross-organizational resources sharing, where each organization uses its own Windows Active Directory. Consider the case in which Organization A and B have some agreement on sharing resources and an employee of Organization A is trying to access some resources of Organization B. That is, Organization B here is a service provider. **1.** The employee first authenticates himself at Organization A. **2.** The employee requests some resources at Organization B. **3.** If the employee is authenticated at Organization A, TrustBridge, which runs on both sides, performs identity federation so that Organization B shares his identity. **4.** Permission is granted if the request is legal based on the agreement between the two organizations. **5.** The employee can now use these resources. (Note that steps 3 and 4 require no actions from the employee.)

Similar to the Liberty infrastructure, TrustBridge allows organizations share resources securely and also requires employees to login just once to access various resources. Therefore, it satisfies the first and third property. In TrustBridge, Windows Active Directory uses Kerberos authentication system to issue tickets to employees. The Kerberos system is stateless as it keeps authentication status in the tickets [19]. As a result, the Windows Active Directory keeps no session states for employees. Therefore, TrustBridge also satisfies the second property. However, TrustBridge does not satisfy our fourth property as the infrastructure is not open to heterogeneous platforms. In TrustBridge, organizations must run Windows Active



(A) client subsystem, (B) business logic subsystem, (C) AAA subsystem and (D) service subsystem

Figure 4. Overview of .TRUST protocols. Note that once an ID card is issued by the authentication server, its information will be included in future message passing.

Directory in order to issue credentials. Such approach imposes burdens to organizations that want to interoperate with others.

In short, while the Liberty infrastructure and TrustBridge provide solutions to resources sharing across organizations, both approaches have drawbacks. In the following section, we introduce a new architecture, called .TRUST, which satisfies all four properties.

3. An overview of the .TRUST architecture

Beyond meeting the four properties, the .TRUST architecture is designed to provide some additional desirable features. For example, service providers are allowed to focus on the services they provide, by leveraging the AAA functionality provided by the .TRUST architecture. Moreover, organizations are able to control their own private information despite sharing specific data with other parties. They will export information on a task-dependent way, i.e., they only export information that is necessary to complete the task at hand.

3.1. Subsystems

The .TRUST architecture consists of four subsystems. These subsystems are client subsystem, business logic subsystem, AAA subsystem and service subsystem, as shown in Figure 4.

- *Client subsystem.* The client subsystem communicates with authentication, business logic, authorization and accounting servers in sequence to obtain enough credentials for its desired service. It then sends the credentials to the service unit to request for authorized services. Upon receiving the request, the service subsystem will deliver the service to the client.
- *Business logic subsystem.* The business logic subsystem maintains all the contracts between companies and services. It also provides support for companies to subscribe or unsubscribe to available services.
- *AAA subsystem.* The AAA subsystem contains authentication, authorization and accounting servers. The authentication server verifies user ID and password and also determines the attributes of the client. The attributes may be demographic in nature such as a human resources recruiter or a student enrolled in a specific course of a college. The attributes of a given identity may also change over time. Therefore, companies must keep the authentication server well informed. The authorization server maintains (attribute, service type) pairs for each company, and determines whether a user with a set of attributes is permitted to use certain services. The accounting server keeps track of service usage of each company.
- *Service subsystem.* The service subsystem provides the service to the client after verifying its request. It also keeps track of service usage for billing.

3.2. Client protocols in .TRUST

Figure 4 illustrates the interactions between these six units in the four subsystems. The .TRUST architecture is entirely Web-based. All the communications are handled by SOAP, an XML-based protocol. All services are specified by UDDI. Packet encryptions based on standards such as IPSec and SSL are used to provide confidentiality and integrity protection to communications among parties.

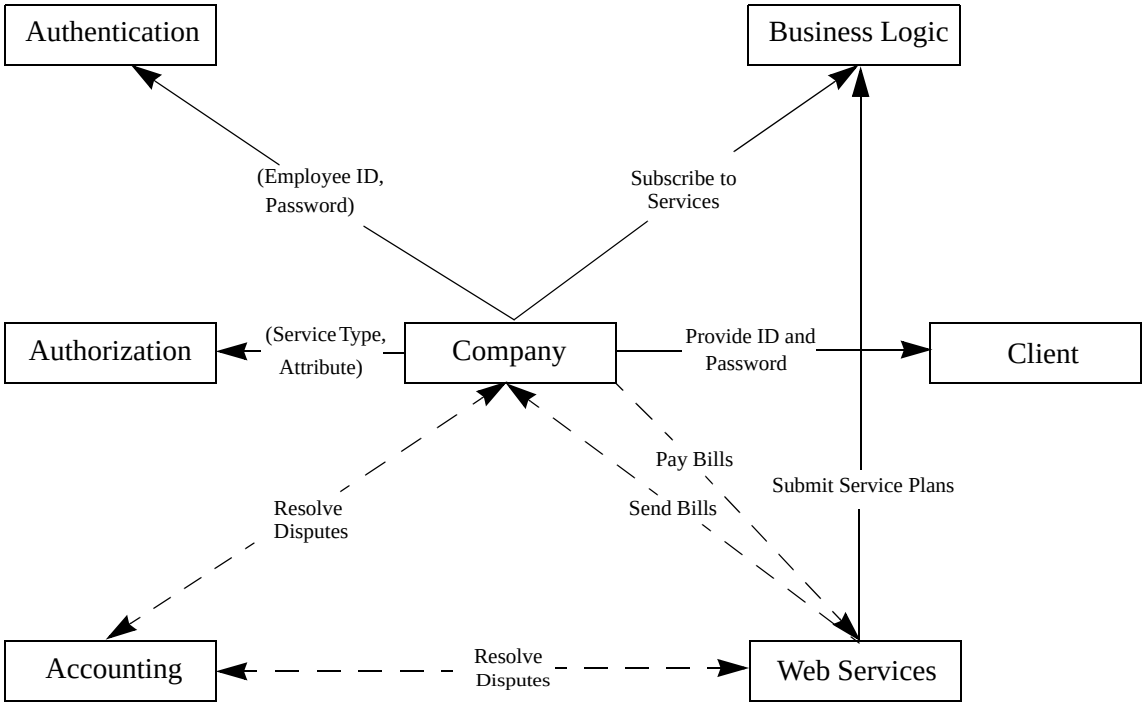


Figure 5. Management protocols of .TRUST.

Note that in practice, there are many exceptional cases we may need to consider. For instance, a Web server may impose limitation on the maximum number of concurrent users from a single company. When the limit is reached, the Web service will send a denial ticket to the client application, and the client application will forward the ticket to the accounting server to cancel the record accordingly.

3.3. Management protocols in .TRUST

We now describe management protocols (See Figure 5) that take place as background communications in the architecture. These protocols can be classified into two groups. The first group consists of the interactions before each unit is ready for service (shown as solid lines in Figure 5). At this stage, each unit needs to acquire certain information in order to provide the service. For instance, the authentication server needs a list of (ID, password) pairs for employees of each company and the business logic server needs a list of service plans from each Web service. The second group consists of the interactions after some services have been provided (shown as dashed lines in Figure 5). For instance, the company needs to pay for the services used and if there is any dispute, the company and the Web services provider will need to consult the accounting server to resolve it.

3.4. Mapping high-level business logic onto architecture rules

It is necessary to facilitate translation of high-level business policies into rules to be executed by the servers of the architecture. For example, the .TRUST architecture provides a method to translate group-based authorization rules into individual-based rules. This supports companies that use authorization polices expressed in terms of services for departments or groups that are authorized to receive. The authorization server and accounting server in the architecture will implement these rules for individuals based on their group membership.

3.5. Comparison with Kerberos

The .TRUST architecture resembles the Kerberos authentication system [20]. In Kerberos, a client authenticates itself to a trusted Key Distribution Center (KDC) and obtains a ticket-granting ticket (TGT). The client system uses this TGT to access ticket-granting service (TGS) and obtains a service ticket. The client then presents this service ticket to the requested network service. The service ticket carries out mutual authentication. That is, it proves both the user's identity to the service and the service's identity to the user. The network service provides services to the client if the authentication is valid [20, 21].

In .TRUST, the authentication server acts as a KDC. The ID card it issues is essentially a TGT. The client presents the ID card when requesting services at the business logic server. Here, the combination of business logic server, authorization server and accounting server serves as a TGS in that an admission ticket is granted only if the service has been subscribed, the client has been authorized to access that service, and necessary record keeping has been

completed. The admission ticket carries out mutual authentication between the client and the service provider.

The .TRUST architecture differs from Kerberos in that it provides additional functionalities which include business logic, authorization and accounting services. The architecture facilitates service subscriptions through the business logic subsystem, and also allows companies to control service usages by imposing various policies at the authorization server. In addition, the accounting server can support arbitration functions to resolve possible payment disputes.

Moreover, we note that in .TRUST each of the AAA functionalities is implemented on a single server and each server can be managed independently. In the next section, we describe the rationales for such design in detail.

4. Design principles and properties

4.1. Design principles

We used three principles in the design of the .TRUST architecture. First, each server in the architecture should perform a well-defined, focused function, and its operation can be managed independently as an individual business unit.

Second, each server should minimize the amount of a company's private information that it requires to complete its duty. In addition, the server should be able to hide the private information using, e.g., anonymous methods.

Third, each server should be stateless in the sense that the architecture does not store state information related to the transaction of each client. This will improve the reliability of the architecture and ease its maintenance.

Based on these design principles, we have made some design choices. The first one concerns whether we should store the (service type, attribute) pairs for each company in the business logic server or in the authorization server. If the business logic server were allowed to contain such information, it could decide the service type that the client will receive based on the ID card transmitted. However, such arrangement would require the business logic server to know more information than necessary. This would be against the principle that the duty of the business logic unit should be solely to maintain the record of contracts between companies, whereas that of the authorization server should be to grant the service type based on the attribute. For these reasons, we chose to store (service type, attribute) pairs in the authorization server.

We also considered an alternative model where the client does not communicate with the business logic server. Instead, once the client is authenticated, it will contact the authorization server with a service request. The authorization server communicates directly with the business logic server with the client's ID card and fetches the service plans information in order to grant the level of services for the client. This model adds unnecessary inter-dependence between the business logic server and the authorization server, and thus is not desirable according to our first design principle.

4.2. Properties of .TRUST

By following these design principles, the .TRUST architecture can achieve high security and simplified system:

- *Server Independence*

  Unlike the Liberty infrastructure in which the identity provider and service providers have to interoperate with each other to provide services, servers involved in .TRUST have no direct communications with each other and thus can be implemented and managed independently. They interact with the client directly by receiving and sending encrypted messages. Because of this independence, different companies can own and manage their own servers.

- *Lowered Security and Privacy Risks*

  The .TRUST architecture can be used to implement task-dependent security and privacy systems. Companies can maximize their control over their private information. They only export information that is necessary to complete the tasks. Furthermore, the separation of tasks ensures that no server has the full knowledge of companies' private information. Therefore, the .TRUST architecture imposes lower risks than traditional approaches in which all private information of a company is stored at centralized servers.

- *Scalability and Reusability*

  Because of the stateless nature of the architecture, the architecture can be scaled up to include multiple copies for each server type to handle increased load. In addition, the same architecture can be used for multiple purposes. For instance, the same system can be used to allow companies to update their authentication or authorization information if we consider the authentication or authorization server itself as a Web service. Figure 6 illustrates the process when a company wants to update its authorization information.

5. Application examples

To prove the concept, we have implemented some simple applications on a laboratory prototype of .TRUST. These applications exemplify some of typical business scenarios encountered today. In these three applications, we assume the Web service provider is PDC, a package delivery company. We only give a full description for scenario 1 as the architecture works in a similar way in the other two scenarios. Notice that although different Web services could be implemented for each of these three scenarios, the .TRUST architecture is general enough to handle all of them.

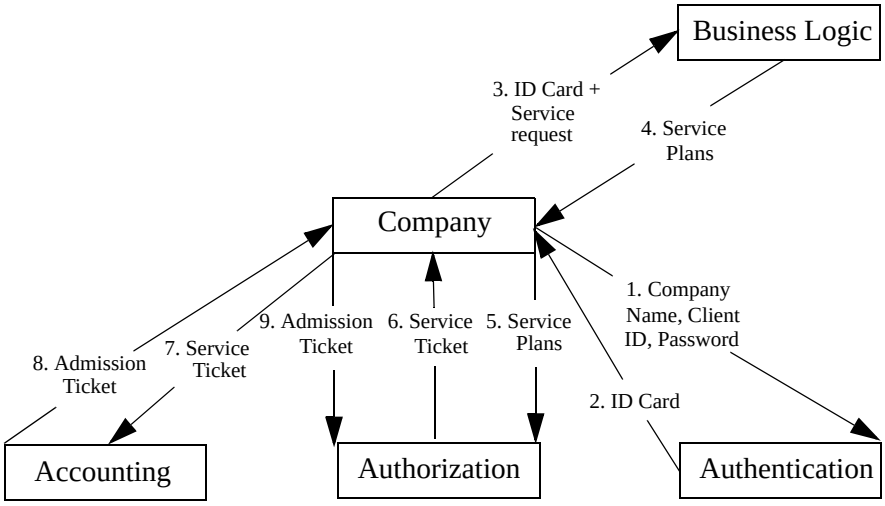


Figure 6. The sequence of interactions for updating authorization information. In this case, the authorization server is the Web service provider.

5.1. Scenario 1

A professor at university X in the east coast is attending a conference at university Y in the west coast. While at Y, he wants to send several packages back to his university X. He then launches a client application to connect to the authentication server. The authentication server checks his login name and password and identifies him as a professor at X. The server then sends him an electronic ID card. The client application forwards the ID card with his service request, in this case shipping packages, to the business logic server. The business logic server checks the contracts between university X and PDC. Suppose that X subscribes to two service plans from PDC: a premiere plan which allows overnight delivery service, and a regular plan which allows 3 Day delivery service. The business logic server sends these two service plans back. Then the client application sends this information to the authorization server. The authorization server maintains service type information for different positions in each company. Assume in this case, it finds out that a professor at X is eligible for the premiere plan. It then sends a service ticket back that includes information such as "University X, Premiere Plan." The professor then sends the service ticket to the accounting server. The accounting server records the service usage and sends an admission ticket to the client. The client then forwards the ticket to the PDC server. The PDC server checks the ticket and allows the professor to send a package using the premiere plan, i.e., by overnight delivery. It also keeps a copy of the ticket for billing purposes.

Note that authorization policies can be implemented on the authorization server and the accounting server. For example, a policy may restrict the number of packages shipped by the premiere plan per day by the same person. Thus after the authorization server determines that the professor is eligible for the premiere plan, it will include the rule in the service ticket. Once the accounting server receives the service ticket, it will execute the rule based on its record about service usages and decide whether to issue the admission ticket.

5.2. Scenario 2

PDC would like to purchase new computers for its employees. For obvious reasons, the choice of computer types and configurations should be limited. PDC makes a contract with a computer manufacturer, which consists of a permitted range of computers. PDC then decides appropriate computer models for each employee position and submits this information to authorization server. Employees of PDC can simply order their computers online using the .TRUST architecture. Because of its generality, the architecture could allow PDC to authorize its employees to purchase bundled systems including computers and printers from multiple vendors.

5.3. Scenario 3

PDC also provides same day delivery service for urgent packages in certain regions. It is likely that by utilizing transportation services from other companies, particularly these local companies, PDC may save both time and money. The .TRUST architecture allows PDC to make use of services from other companies more easily. As a result, PDC may improve and expand its services and thus attract more business.

6. Conclusion

In this paper we have described a stateless AAA architecture, called .TRUST, to achieve resources sharing between organizations. We consider our system an improvement over the two infrastructures designed by

Microsoft and the Liberty Alliance in that our architecture satisfies all four properties outlined in Section 1 and possesses several other desirable features.

First, the .TRUST architecture allows organizations to choose any appropriate security schemes to accomplish secure resources sharing. Second, owing to its stateless nature, the architecture can be scaled up easily to support a large number of clients and services, and in the meantime, stay resilient in defense again DoS attacks. Third, as the AAA servers in .TRUST are separated from each individual service provider, a user only needs to remember one (login name, password) pair in order to gain access to various services. Fourth, .TRUST is entirely Web-based and can work across any platforms with standard Web functionalities. In addition, service providers in .TRUST are no longer burdened by implementing and maintaining authentication and authorization functionalities and can focus on the services they are providing. Finally, .TRUST allows organizations to control their own private information and export information only if it is necessary to complete the task.

We have also described three application examples to illustrate how the .TRUST architecture can help exploit trust-based relationships among businesses in a secure and flexible way. These application examples have been implemented on a laboratory prototype of .TRUST.

Acknowledgment

This work was supported in part by an E-Business Research & Curriculum Laboratory Grant from Intel Corporation and in part by DARPA through AFRL/IFKD under contract F33615-01-C-1983.

References

[1] Lynch, C., *A white paper on authentication and access management issues incross-organizational use of networked information resources*, Coalition for Networked Information, April 1998. http://www.cni.org/projects/authentication/authentication-wp.html

[2] Levijoki, S., *Authentication, authorization and accounting in ad hoc networks*, May 2000. http://www.tml.hut.fi/Opinnot/Tik-110.551/2000/papers/authentication/aaa.htm

[3] Vollbrecht, J., et al., *RFC 2905, "AAA authorization application examples"*, August 2000. ftp://ftp.isi.edu/in-notes/rfc2905.txt

[4] *Novell simplifies, personalizes net experience with Novell Portal Services*, press release, March 19, 2001. http://www.novell.com/news/press/archive/2001/03/pr01023.html

[5] Haught, D., *Designing commercial Web services: Part I - security and flexibility with the .NET framework*, http://www.aspnetpro.com/features/2002/01/asp200201dh_f/asp200201dh_f.asp

[6] Costello, S., *Researcher: DDoS attacks are growing threats*, IDG News Service, June 2001. http://www.nwfusion.com/news/2001/0606ddos.html

[7] MacVittie, L., *Microsoft keys in on the enterprise*, March 13, 2002. http://www.networkcomputing.com/1310/1310buzz1.html

[8] *Securing the broadband cable network*, RSA e-security Today, Vol. 2, No. 4, October 2001. http://www.rsasecurity.com/newsletter/v2n4/broadband.html

[9] *Liberty architecture overview version 1.1*, January 15, 2003. http://www.projectliberty.org/specs/liberty-architecture-overview-v1.1.pdf

[10] Westmacott, I., *The stateful Web*, Computer Publishing Group, 1998. http://webserver.cpg.com/features/cover/3.5/

[11] *Cisco AAA case study overview*. http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/secsols/aaasols/c262c1.htm - 2418

[12] *System authentication plug-ins*, Windows Media Services. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmsrvsdk/htm/wmsauthenticationplugins.asp

[13] *.NET Passport: Balanced authentication solutions*, Microsoft Corporation. http://www.microsoft.com/net/downloads/net_passport.doc

[14] *Microsoft's federated security and identity roadmap*, Microsoft Corporation, June 2002. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/wsfederate.asp

[15] Glass, B., *Is Microsoft's "TrustBridge" really "HackerBridge"?*. http://www.extremetech.com/article2/0,3973,55154,00.asp

[16] Kaneshige, T., *Microsoft's TrustBridge reveals roadmap for Web-services security products; analyst wonders whether Microsoft can truly be 'trusted'*. http://www.line56.com/articles/default.asp?ArticleID=3735

[17] Fontana, J., *Microsoft touts tighter Web services security.* http://www.nwfusion.com/news/2002/0610microsoft.html

[18] *Microsoft Windows "TrustBridge" to enable organizations to share user identities across business boundaries*, June 6, 2002. http://www.microsoft.com/presspass/press/2002/Jun02/06-06TrustbridgePR.asp

[19] *Windows 2000 authentication.* http://www.windowsitlibrary.com/Content/617/06/2.html

[20] *Kerberos: The Network Authentication Protocol.* http://web.mit.edu/kerberos/www/

[21] *Kerberos V5 authentication.* http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windowsserver2003/proddocs/standard/sag_SEconceptsUnAuthKerb.asp