Security in a Web Services World: A Proposed Architecture and Roadmap

A joint security whitepaper from IBM Corporation and Microsoft Corporation. April 7, 2002, Version 1.0

April 2002

This document describes a proposed strategy for addressing security within a Web service environment. It defines a comprehensive Web service security model that supports, integrates and unifies several popular security models, mechanisms, and technologies (including both symmetric and public key technologies) in a way that enables a variety of systems to securely interoperate in a platform- and language-neutral manner. It also describes a set of specifications and scenarios that show how these specifications might be used together.

This is a preliminary document and may be changed substantially over time. The information contained in this document represents the current view of International Business Machine and Microsoft Corporation on the issues discussed as of the date of publication. Because IBM and Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of IBM and Microsoft, and IBM and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

The presentation, distribution or other dissemination of the information contained in this document is not a license, either expressly or impliedly, to any intellectual property owned or controlled by IBM or Microsoft and\or any other third party. IBM, Microsoft and\or any other third party may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you any license to IBM's or Microsoft's or any other third party's patents, trademarks, copyrights, or other intellectual property. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, places, or events is intended or should be inferred.

This document and the information contained herein is provided on an "AS IS" basis and to the maximum extent permitted by applicable law, IBM and Microsoft provides the document AS IS AND WITH ALL FAULTS, and hereby disclaims all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence, all with regard to the document. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO THE DOCUMENT.

Contents:

Related content:

Web Services Security
Subscribe to the developerWorks newsletter
More dW Security resources

Also in the Web services zone:

Tutorials
Tools and products
Articles

Executive Summary

The IT industry has been talking about Web services for almost two years. The benefits of having a loosely-coupled, language-neutral, platform-independent way of linking applications within organizations, across enterprises, and across the Internet are becoming more evident as Web services are used in pilot programs and in wide-scale production. Moving forward, our customers, industry analysts, and the press identify a key area that needs to be addressed as Web services become more mainstream: security. This document proposes a technical strategy and roadmap whereby the industry can produce and implement a standards-based architecture that is comprehensive yet flexible enough to meet the Web services security needs of real businesses.

A key benefit of the emerging Web services architecture is the ability to deliver integrated, interoperable solutions. Ensuring the integrity, confidentiality and security of Web services through the application of a comprehensive security model is critical, both for organizations and their customers.

Responding to concerns expressed both from our customers and the industry, IBM and Microsoft have collaborated on this proposed Web services security plan and roadmap for developing a set of Web Service Security specifications that address how to provide protection for messages exchanged in a Web service environment.

For the first time, we have created a security model that brings together formerly incompatible security technologies such as public key infrastructure, Kerberos, and others. In short, this is not an idealized framework but a practical one that can allow us to build secure Web services in the heterogeneous IT world in which our customers live today.

In this document we present a broad set of specifications that cover security technologies including authentication, authorization, privacy, trust, integrity, confidentiality, secure communications channels, federation, delegation and auditing across a wide spectrum of application and business topologies. These specifications provide a framework that is extensible, flexible, and maximizes existing investments in security infrastructure. These specifications subsume and expand upon the ideas expressed in similar specifications previously proposed by IBM and Microsoft (namely the SOAP-Security, WS-Security and WS-License specifications).

By leveraging the natural extensibility that is at the core of the Web services model, the specifications build upon foundational technologies such as SOAP, WSDL, XML Digital Signatures, XML Encryption and SSL/TLS. This allows Web service providers and requesters to develop solutions that meet the individual security requirements of their applications.

IBM and Microsoft intend to work with customers, partners and standards bodies to evolve and improve upon this security model in a phased approach. We are seeding this effort with the WS-Security specification. WS-Security defines the core facilities for protecting the integrity and confidentiality of a message, as well as mechanisms for associating security-related claims with the message. While WS-Security is the cornerstone of this effort, it is only the beginning and we will cooperate with the industry to produce additional specifications that will deal with policy, trust and privacy issues.

To make the issues and solutions discussed in this document as concrete as possible, we discuss several scenarios that reflect current and anticipated applications of web services. These include firewall processing, privacy, use of browser and mobile clients, access control, delegation, and auditing.

We anticipate concerns about what can be done to ensure interoperability and consistent implementation of the various proposed specifications. To address this, IBM and Microsoft will work closely with standards organizations, the developer community, and with industry organizations such as WS-I.org to develop interoperability profiles and tests that will provide guidance to tool vendors.

This document outlines a comprehensive, modular solution that, when implemented, will allow customers to build interoperable and secure Web services that leverage and expand upon existing investments in security infrastructure while allowing them to take full advantage of the integration and interoperability benefits Web service technologies have to offer.

Introduction and Motivation

Providing a comprehensive model of security functions and components for Web services requires the integration of currently available processes and technologies with the evolving security requirements of future applications. It demands unifying concepts; it requires solutions to both technological (secure messaging) and business process (policy, risk, trust) issues; and finally, it requires coordinated efforts by platform vendors, application developers, network and infrastructure providers, and customers.

Unifying the range of security technologies available means that the functional requirements of application security must be abstracted from specific mechanisms employed. For example, a customer making an on-line purchase should not be impacted by whether they are using a cell phone or a laptop computer, as long as each device can securely express the proper identity.

The goal is to enable customers to easily build interoperable solutions using heterogeneous systems. For instance, the secure messaging model proposed later in this document supports both public key infrastructure (PKI) and Kerberos identity mechanisms as particular embodiments of a more-general facility and is capable of being extended to support additional security mechanisms. Integration through the abstractions of a single security model enables organizations to use their existing investments in security technologies while communicating with organizations using different technologies.

Further, as organizations using different identity mechanisms collaborate using Web services, the security trust model provides a flexible framework within which the organizations can interconnect when configured with appropriate authorization.

At the same time, every customer and every Web service has its own unique security requirements based upon their particular business needs and operational environment. Within workgroup settings, for instance, simplicity and ease of operations are a top concern, while for public Internet applications the ability to withstand concerted denial-of-service attacks is a higher priority. Because these requirements may be combined in many ways and expressed at different levels of specificity, a successful approach to Web service security requires a set of flexible, interoperable security primitives that, through policy and configuration, enable a variety of secure solutions.

To address these challenges, this document proposes an evolutionary approach to creating secure, interoperable Web services based on a set of security abstractions that unify formerly dissimilar technologies. This enables specialization to particular customer requirements within an overall framework while at the same time permitting technologies to evolve over time and be incrementally deployed.

As an example of this evolutionary approach, the secure messaging model can be added to existing transport-level security solutions. A customer can add message-level integrity or persistent confidentiality (encryption of message elements) to an existing Web service whose messages are carried through, for example, Secure Sockets Layer (SSL/TLS). The messages now have integrity (or confidentiality) that persists beyond the transport layer.

We anticipate that the proposed model and specifications that emerge will be broadly available from multiple vendors and will be considered by appropriate standards organizations. Together, the model, specifications, and standards process enable businesses to quickly and cost-effectively increase the security of their existing applications and to confidently develop new interoperable, secure Web services.

The business advantage of such a model is clear. By framing a comprehensive security architecture for Web services, organizations and customers can be better ensured that their investments and assets are protected as business processes become increasingly recast as Web services.

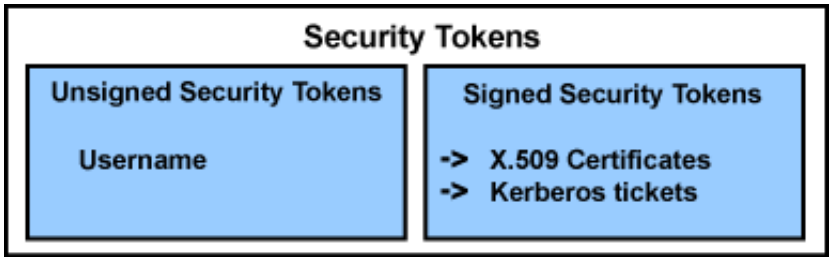
Web Services Security Model Terminology

Because terminology varies between technologies, this document defines several terms that may be applied consistently across the different security formats and mechanisms. Consequently, the terminology used here may be different from other specifications and is defined so that the reader can map the terms to their preferred vocabulary.

- **Web service** -- The term "Web service" is broadly applicable to a wide variety of network based application topologies. In this document, we use the term "Web service" to describe application components whose functionality and interfaces are exposed to potential users through the application of existing and emerging Web technology standards including XML, SOAP, WSDL, and HTTP. In contrast to Web sites, browser-based interactions or platform-dependent technologies, Web services are services offered computer-to-computer, via defined formats and protocols, in a platform-independent and language-neutral manner.
- **Security Token** -- We define a security token as a representation of security-related information (e.g. X.509 certificate, Kerberos tickets and authenticators, mobile device security tokens from SIM cards, username, etc.).

  The following diagram shows some of the different kinds of security tokens.

  Figure 1. Different kinds of security tokens



- **Signed Security Token** -- We define a *signed security token* as a security token that contains a set of related claims (assertions) cryptographically endorsed by an issuer. Examples of signed security tokens include X.509 certificates and Kerberos tickets.
- **Claims** -- A claim is a statement about a subject either by the subject or by an relying party that associates the subject with the claim. An important point is that this specification does not attempt to limit the types of claims that can be made, nor does it attempt to limit how these claims may be expressed. Claims can be about keys potentially used to sign or encrypt messages. Claims can be statements the security token conveys. Claims may be used, for example, to assert the senders identity or an authorized role.
- **Subject** -- The subject of the security token is a principal (e.g. a person, an application or a business entity) about which the claims expressed in the security token apply. Specifically, the subject, as the owner of the security token possesses information necessary to prove ownership of the security token.
- **Proof-of-Possession** -- We define *proof-of-possession* to be information used in the process of proving ownership of a security token or set of claims. For example, proof-of-possession might be the private key associated with a security token that contains a public key.
- **Web Service Endpoint Policy** -- Web services have complete flexibility in specifying the claims they require in order to process messages. Collectively we refer to these required claims and related information as the "Web Service Endpoint Policy". Endpoint policies may be expressed in XML and can be used to indicate requirements related to authentication (e.g. proof of user or group identity), authorization (e.g. proof of certain execution capabilities), or other custom requirements.
- **Claim Requirements** -- Claim requirements can be tied to whole messages or elements of messages, to all actions of a given type or to actions only under certain circumstances. For example, a service may require a requestor to prove authority for purchase amounts greater than a stated limit.
- **Intermediaries** -- As SOAP messages are sent from an initial requester to a service, they may be operated on by intermediaries that perform actions such as routing the message or even modifying the message. For example, an intermediary may add headers, encrypt or decrypt pieces of the message, or add additional security tokens. In such situations, care should be taken so that alterations to the message do not invalidate message integrity, violate the trust model, or destroy accountability.
- **Actor** -- An *actor* is an intermediary or endpoint (as defined in the SOAP specification) which is identified by a URI and which processes a SOAP message. Neither users NOR client software (e.g. browsers) are actors.

Web Services Security Model Principles

Web services can be accessed by sending SOAP messages to service endpoints identified by URIs, requesting specific actions, and receiving SOAP message responses (including fault indications). Within this context, the broad goal of securing Web services breaks into the subsidiary goals of providing facilities for securing the integrity and confidentiality of the messages and for ensuring that the service acts only on requests in messages that express the claims required by policies.

Today the Secure Socket Layer (SSL) along with the de facto Transport Layer Security (TLS) is used to provide transport level security for web services applications. SSL/TLS offers several security features including authentication, data integrity and data confidentiality. SSL/TLS enables point-to-point secure sessions.

IPSec is another network layer standard for transport security that may become important for Web services. Like SSL/TLS, IPSec also provides secure sessions with host authentication, data integrity and data confidentiality.

Today's Web service application topologies include a broad combination of mobile devices, gateways, proxies, load balancers, demilitarized zones (DMZs), outsourced data centers, and globally distributed, dynamically configured systems. All of these systems rely on the ability for message processing intermediaries to forward messages. Specifically, the SOAP message model operates on logical endpoints that abstract the physical network and application infrastructure and therefore frequently incorporates a multi-hop topology with intermediate actors.

When data is received and forwarded on by an intermediary beyond the transport layer both the integrity of data and any security information that flows with it maybe lost This forces any upstream message processors to rely on the security evaluations made by previous intermediaries and to completely trust their handling of the content of messages." What is needed in a comprehensive Web service security architecture is a mechanism that provides end-to-end security. Successful Web service security solutions will be able to leverage both transport and application layer security mechanisms to provide a comprehensive suite of security capabilities.

Figure 2. Point-to-point configuration

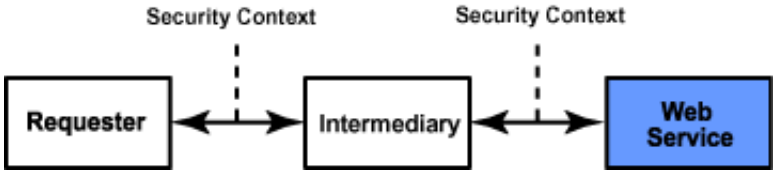
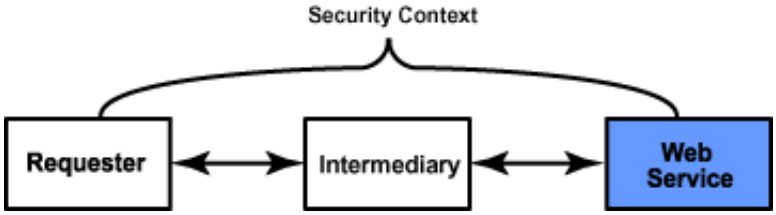


Figure 3. End-to-end configuration



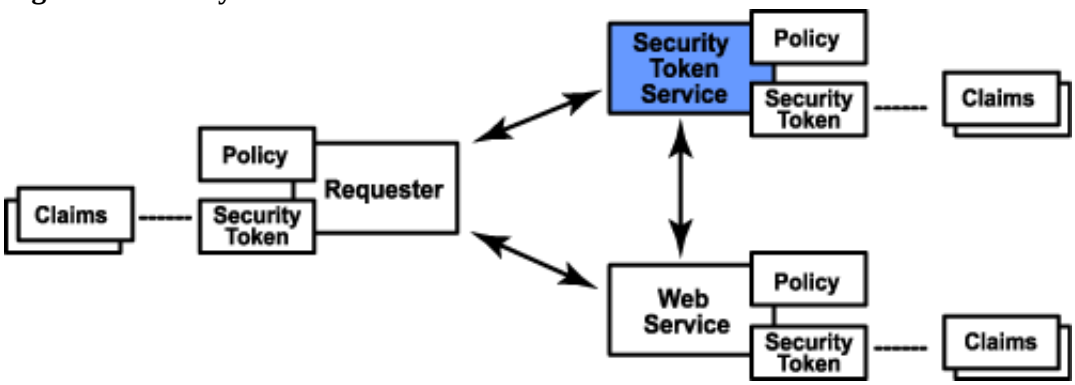
The Web service security model described herein enables us to achieve these goals by a process in which:

- A Web service can require that an incoming message prove a set of *claims* (e.g., name, key, permission, capability, etc.). If a message arrives without having the required claims, the service may ignore or reject the message. We refer to the set of required claims and related information as *policy*.
- A requester can send messages with proof of the required claims by associating *security tokens* with the messages. Thus, messages both demand a specific action and prove that their sender has the claim to demand the action.
- When a requester does not have the required claims, the requester or someone on its behalf can try to obtain the necessary claims by contacting other Web services. These other Web services, which we refer to as *security token services*, may in turn require their own set of claims. Security token services broker trust between different trust domains by issuing security tokens.

This model is illustrated in the figure below, showing that any requester may also be a service, and that the

Security Token Service may also fully be a Web service, including expressing policy and requiring security tokens.

Figure 4. Security token service model

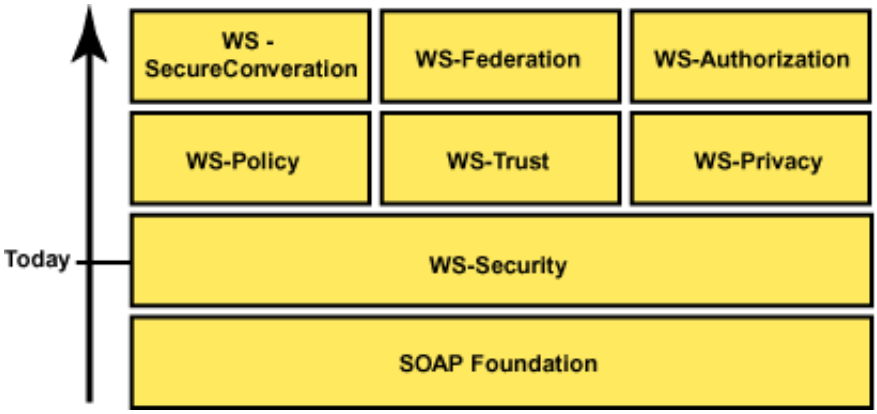


This general messaging model -- claims, policies and security tokens -- subsumes and supports several more specific models such as identity-based-security, access control lists, and capabilities-based-security. It allows use of existing technologies such as X.509 public-key certificates, Kerberos shared-secret tickets and even password digests. As we will discuss later, it also provides an integrating abstraction allowing systems to build a bridge between different security technologies. The general model is sufficient to construct higher-level key exchange, authentication, authorization, auditing, and trust mechanisms.

Web Services Security Specifications
The security strategy expressed here and the WS-Security specification introduced below provide the strategic goals and cornerstone for this proposed Web services security model.

Moving forward, we are continuing the process of working with customers, partners and standards organizations to develop an initial set of Web service security specifications.

Figure 5. Web Services Security Specifications



This set will include a message security model (WS-Security) along with a Web service endpoint policy (WS-Policy), a trust model (WS-Trust), and a privacy model (WS-Privacy). Together these initial specifications provide the foundation upon which we can work to establish secure interoperable Web services across trust domains.

Building on these initial specifications we will continue to work with customers, partners and standards organizations to provide follow-on specifications for secure conversations (WS-SecureConversation), federated trust (WS-Federation), and authorization (WS-Authorization).

The combination of these security specifications enable many scenarios (some of which are described later in this document) that are difficult to implement with today's more basic security mechanisms.

In parallel, we will propose and move forward with related activities that enhance the Web services security framework with specifications related to auditing, management, and privacy.

Additionally, IBM and Microsoft are committed to working with organizations like WS-I on interoperability profiles.

The combination of security specifications, related activities, and interoperability profiles will enable customers to easily build interoperable secure Web services.

Each of the proposed specifications is summarized below:

Initial Specifications

- WS-Security: describes how to attach signature and encryption headers to SOAP messages. In addition, it describes how to attach security tokens, including binary security tokens such as X.509 certificates and Kerberos tickets, to messages.
- WS-Policy: will describe the capabilities and constraints of the security (and other business) policies on intermediaries and endpoints (e.g. required security tokens, supported encryption algorithms, privacy rules).
- WS-Trust: will describe a framework for trust models that enables Web services to securely interoperate.
- WS-Privacy: will describe a model for how Web services and requesters state subject privacy preferences and organizational privacy practice statements..

Follow-On Specifications

- WS-SecureConversation: will describe how to manage and authenticate message exchanges between parties including security context exchange and establishing and deriving session keys.
- WS-Federation: will describe how to manage and broker the trust relationships in a heterogeneous federated environment including support for federated identities.
- WS-Authorization: will describe how to manage authorization data and authorization policies.

Providing security for Web Services requires due diligence in the production of the descriptions, specification and profiles in a number of functional areas. These documents will change and evolve through a process that balances the needs of customers with the needs of the Web services development community and our own educational process as we go through the specification process.

WS-Security
WS-Security describes enhancements to SOAP messaging to provide *quality of protection* through message integrity and message confidentiality. As well, this specification defines how to attach and include security tokens within SOAP messages. Finally, a mechanism is provided for specifying binary encoded security tokens (e.g. X.509 certificates). These mechanisms can be used independently or in combination to accommodate a wide variety of security models and encryption technologies.

WS-Security provides a general-purpose mechanism for associating security tokens with messages. No specific type of security token is required by WS-Security. It is designed to be extensible (e.g. support multiple security token formats). For example, a requester might provide proof of identity and proof that they have a particular business certification.

Message integrity is provided by leveraging XML Signature in conjunction with security tokens (which may contain or imply key data) to ensure that messages are transmitted without modifications. The integrity mechanisms are designed to support multiple signatures, potentially by multiple actors, and to be extensible to support additional signature formats. The signatures may reference (i.e. point to) a security token.

Similarly, message confidentiality is provided by leveraging XML Encryption in conjunction with security tokens to keep portions of SOAP messages confidential. The encryption mechanisms are designed to support additional encryption technologies, processes, and operations by multiple actors. The encryption may also reference a security token.

Finally, WS-Security describes a mechanism for encoding binary security tokens. Specifically, the specification describes how to encode X.509 certificates and Kerberos tickets as well as how to include opaque encrypted keys. It also includes extensibility mechanisms that can be used to further describe the characteristics of the security tokens that are included with a message.

WS-Policy

WS-Policy will describe how senders and receivers can specify their requirements and capabilities.

WS-Policy will be fully extensible and will not place limits on the types of requirements and capabilities that may be described; however, the specification will likely identify several basic service attributes including privacy attributes, encoding formats, security token requirements, and supported algorithms.

This specification will define a generic SOAP policy format, which can support more than just security policies. This specification will also define a mechanism for attaching service policies to SOAP messages.

WS-Trust
WS-Trust will describe the model for establishing both direct and brokered trust relationships (including third parties and intermediaries).

This specification will describe how existing direct trust relationships may be used as the basis for brokering trust through the creation of security token issuance services. These security token issuance services build on WS-Security to transfer the requisite security tokens in a manner that ensures the integrity and confidentiality of those tokens.

This specification then will describe how several existing trust mechanisms may be used in conjunction with this trust model.

Finally, the trust model will explicitly allows for, but will not mandate, delegation and impersonation by principals. Note that delegation is consistent with impersonation, but provides additional levels of traceability.

WS-Privacy
Organizations creating, managing, and using Web services will often need to state their privacy policies and require that incoming requests make claims about the senders' adherence to these policies.

By using a combination of WS-Policy, WS-Security, and WS-Trust, organizations can state and indicate conformance to stated privacy policies. This specification will describe a model for how a privacy language may be embedded into WS-Policy descriptions and how WS-Security may be used to associate privacy claims with a message. Finally, this specification will describe how WS-Trust mechanisms can be used to evaluate these privacy claims for both user preferences and organizational practice claims.

WS-SecureConversation
WS-SecureConversation will describe how a Web service can authenticate requester messages, how requesters can authenticate services, and how to establish mutually authenticated security contexts.

This specification will describe how to establish session keys, derived keys, and per-message keys.

Finally, this specification will describe how a service can securely exchange context (collections of claims about security attributes and related data). In order to accomplish this, the specification will describe and build upon the concepts of security token issuance and exchange mechanisms defined in WS-Security and WS-Trust. Using these mechanisms a service might, for example, support security tokens using weak symmetric key technology as well as issue stronger security tokens using non-shared (asymmetric) keys.

WS-SecureConversation is designed to operate at the SOAP message layer so that the messages may traverse a variety of transports and intermediaries. This does not preclude its use within other messaging frameworks. In order to further increase the security of the systems, transport level security may be used in conjunction with both WS-Security and WS-SecureConversation across selected links.

WS-Federation
This specification will define how to construct federated trust scenarios using the WS-Security, WS-Policy, WS-Trust, and WS-SecureConversation specifications. For example, it will describe how to federate Kerberos and PKI infrastructures (as described in the scenarios below).

As well, a trust policy is introduced to indicate and constrain and identify the type of trust that is being brokered.

This specification also will define mechanisms for managing the trust relationships.

WS-Authorization

This specification will describe how access policies for a Web service are specified and managed. In particular it will describe how claims may be specified within security tokens and how these claims will be interpreted at the endpoint.

This specification will be designed to be flexible and extensible with respect to both authorization format and authorization language. This enables the widest range of scenarios and ensures the long-term viability of the security framework.

Relating Web Services Security to Today's Security Models

This Web services security model is compatible with the existing security models for authentication, data integrity and data confidentiality in common use today. As a consequence, it is possible to integrate Web services-based solutions with other existing security models:

- *Transport Security* -- Existing technologies such as secure sockets (SSL/TLS) can provide simple point-to-point integrity and confidentiality for a message. The Web Services security model supports using these existing secure transport mechanisms in conjunction with WS-Security (and other specifications) to provide end-to-end integrity and confidentially in particular across multiple transports, intermediaries, and transmission protocols.
- *PKI* -- At a high level, the PKI model involves certificate authorities issuing certificates with public asymmetric keys and authorities which assert properties other than key ownership (for example, attribute authorities). Owners of such certificates may use the associated keys to express a variety of claims, including identity. The Web services security model supports security token services issuing security tokens using public asymmetric keys. PKI is used here in the broadest sense and does not assume any particular hierarchy or model.
- *Kerberos* -- The Kerberos model relies on communication with the Key Distribution Center (KDC) to broker trust between parties by issuing symmetric keys encrypted for both parties and "introducing" them to one another. The Web services model, again, builds upon the core model with security token services brokering trust by issuing security tokens with encrypted symmetric keys and encrypted testaments.

Note that while the models are compatible, to ensure interoperability, adaptors and/or common algorithms for signatures and encryption will need to be agreed upon or developed.

Existing models for federation, authorization (including delegation), privacy and trust are less common and more ad-hoc. Specifications to address these security properties are identified in the later phases of the strategy.

Often the existing trust models are based on business agreements. An example of this is the UDDI Web service. In UDDI, there are several participants who provide a Universal Business Registry through an agreement to support a set of APIs. Rather than defining a single model for "trust" through the requirement of a specific authentication mechanism, the "trust model" in UDDI gives the responsibility for authentication to the node, which is the custodian of the information. That is, each implementation of the UDDI Web service has its own authentication mechanism and enforces its own access control policy. The "trust" is between operators, and between the requester and the operator that is the custodian of its information.

Scenarios

Below we present a number of scenarios that exemplify how we envision the proposed Web Service security specifications being used. These scenarios are intentionally focused on the technical details to illustrate the capabilities of the overall security strategy. There will be companion documentation that provides detailed business use scenarios.

The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, places, or events is intended or should be inferred.

The list below briefly introduces some of the scenarios that can be supported by the proposed initial specifications and associated deliverables:

- Direct Trust using Username/Password and Transport-Level Security -- This scenario illustrates authentication using a username and password with transport security.

- Direct Trust using Security Tokens -- This scenario illustrates direct trust using X.509 certification and Kerberos service tickets (ST).
- Security Token Acquisition -- This scenario illustrates authentication using a security token stored independently from the message.
- Firewall Processing -- This scenario illustrates how firewalls can leverage this security model for greater degrees of control.
- Issued Security Token -- This scenario illustrates basic authentication.
- Enforcing Business Policy -- This scenario illustrates how to use security token issuance for codifying business processes.
- Privacy -- This scenario illustrates how clients and services can communicate their privacy policies.
- Web Clients -- This scenario illustrates the use of a Web browser as a client.
- Mobile Clients -- This scenario illustrates how mobile clients can securely interact with Web services.

The second set of scenarios is more sophisticated. These scenarios CAN be built on the current deliverables but are in need of follow-on specifications (like WS-SecureConversation) to make interoperability seamless.

- Enabling Federation -- This section describes several different scenarios involving federated trust.
- Validation Service describes how to use a service that validates the security of a message (e.g. signature).
- Supporting Delegation -- This illustrates how to use security tokens for delegation.
- Access Control -- This illustrates how Web services security supports traditional access control list-based security.
- Auditing -- This illustrates the use of auditing to track security-related activities and incidents.

Note that in the descriptions below the use of the term requester is used to describe the broad variety of potential users of a Web Service and is not meant to limit the characteristics of the requester. Requestors can include business entities interacting with a service within a B2B environment or individuals accessing services from a browser or mobile device.

In the figures below, the "blue" boxes represent services and the "light blue" boxes represent security tokens and their identity and delegation claims.

Direct Trust using Username/Password and Transport-Level Security

[1] Here is a very basic example showing how Web services Security can be used with existing transport security mechanisms:

Figure 6: Web services security and existing transport security mechanisms

The requester opens a connection to the Web service using a secure transport (e.g. SSL/TLS). It sends its request and includes a security token that contains its username and password. The service authenticates the information, processes the request, and returns the result.

In this scenario, the message confidentiality and integrity are handled using existing transport security mechanisms.

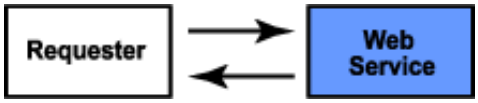
This scenario assumes that the two parties have already used some mechanism to establish a shared secret - the requester password. No assumption is made about the organizational relationship between these parties.

Direct Trust using Security Tokens

[2] This scenario illustrates the use of a security token that is directly trusted by a Web service. Here direct trust means that the requester's security token (or its signing authority) is known and trusted by the Web

service. This scenario assumes that the two parties have used some mechanism to establish a trust relationship for use of the security token. This trust may be established manually, by configuring the application, or by using a secure transport to exchange keys. By secure transport of keys we mean that a transport such as SSL (or other mechanism or process) can be used as a way for a trusted party to assert the validity of a key or security token to a recipient party. No assumption is made about the organizational relationship between the parties.

Figure 7. Direct Trust between two parties



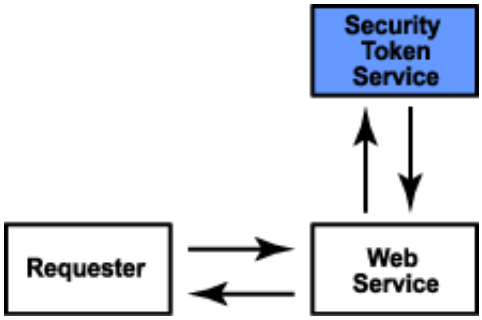
The requester sends a message to a service and includes a signed security token and provides proof-of-possession of the security token using, for example, a signature. The service verifies the proof and evaluates the security token. The signature on the security token is valid and is directly trusted by the service. The service processes the request and returns a result.

Direct trust assumes that the policies for privacy are well understood by the parties involved.

Security Token Acquisition

[3] In some cases, the security token used isn't passed as part of the message. Instead, a security token reference is provided that can be used to locate and acquire the token.

Figure 8. Security tokens by reference



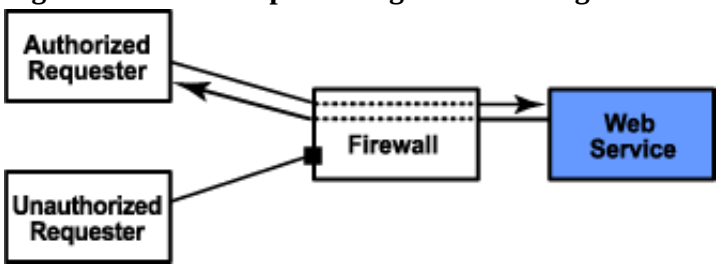
The requester issues a request to the service and includes a reference to the security token and provides proof-of-possession. The Web service uses the provided information to obtain the security token from the token store service and validate the proof. The Web service trusts (note that trust was established outside of the message semantics) the security token, so the request is processed and the response is returned.

Firewall Processing

[4] Firewalls remain a critical component of the Web services security architecture -- they must be able to continue to enforce boundary processing rules.

As shown below, the firewall examines incoming SOAP messages and only allows those from "authorized" requesters to penetrate the firewall.

Figure 9. A Firewall processing SOAP messages



In this scenario we have two requesters sending messages to a Web service inside a firewall. The firewall examines the messages to determine if the requester is "authorized" to send messages to the specified Web service inside the firewall.

In this scenario, the firewall makes this decision by examining the security token used to sign the message. If the signature is valid, and the signing authority for the security token is trusted to authorize messages

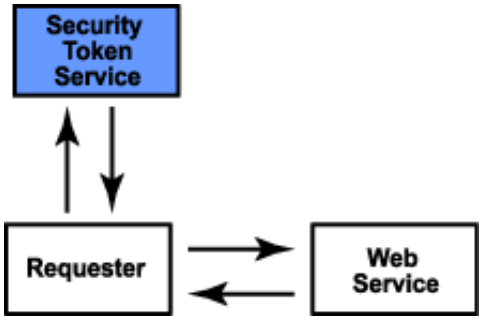
into the firewall, and the token says that it does authorize messages into the firewall, then the message is allowed; otherwise it is rejected. In some cases, a signature may specifically reference the firewall as a SOAP actor.

In other scenarios the firewall could also act as a security token issuing authority and only allow messages that include proof-of-possession of a security token issued by the firewall.

Issued Security Token

5 Here is an example showing how Web services Security supports simple authentication by a trusted party:

Figure 10. Simple authentication by a trusted party



In the first two steps, the requester communicates with a certifying authority to obtain a security token, specifically a signed statement of assertions attesting to the requester's identity.

The requester obtained a security token because the Web service has a policy requiring that a security token of the appropriate type was needed (in this case, proof of identity).

The requester next sends a message, with the security token and a proof of possession attached to the message (think authenticator or signed challenge), to the Web service and receives a response.

Note that in the above description, the operation of the identity certification service was not described in detail. To obtain an identity security token, requesters may use existing security protocols or they may leverage the Web services security specifications.

If we assume the requester is using the Web services security specifications, then the system would operate as follows: the identity service describes its requirements in a policy, and the requester can then request a security token along with its proof of identity.

Note that the identity service is just a security token service. Moreover, the symmetry of Web services allows for any Web service to also encapsulate a security token service.

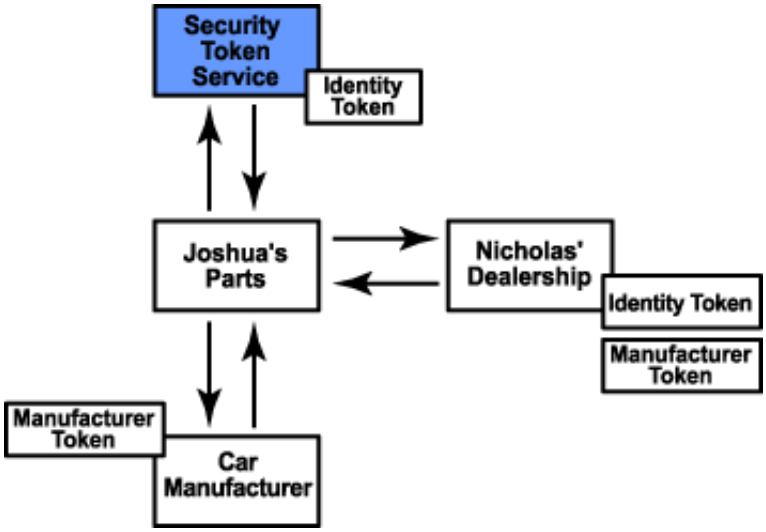
Finally, note that Web services can obtain security tokens on the requester's behalf (from a security token service) and return them to the requester for use on subsequent messages.

Enforcing Business Policy

6 In many business processes there are specific policies that must be enforced. For example, a service may require that consumers have a certain rating or standing with a specific auditing company. With Web services many of these policies can be codified and validated automatically, simplifying the overall process.

Consider the following example:

Figure 11. An example in enforcing policies



Nicholas' Dealership has a Web service that it uses for interacting with its parts suppliers. However, they only want to deal with suppliers whose parts are certified by the manufacturer.

A parts company, Joshua's Parts, would go to the manufacturer and present (and prove) their identity security token (from, for example, the illustrated Security Token Service) and request a security token from the manufacturer stating they are a certified parts dealer (assuming they are certified and in good standing). Joshua's Parts could then contact Nicholas' Dealership and provide (and prove) both security tokens.

If Nicholas' Dealership has codified their business policy into the service policy, the burden of policy conformance could be front-loaded to the parts company (e.g. Joshua's Parts).

The service policy also may specify constraints on what information the manufacturer would be allowed to store to ensure compliance with privacy policy.

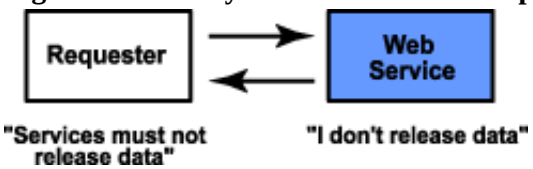
Privacy

[7] Privacy includes a broad set of concerns and will need to be accounted for in each of the specifications that emerge from this strategy.

Basic privacy issues will be addressed by providing privacy statements within the service policy. More sophisticated scenarios, involving delegation and authorization, will be covered in specifications specific to those scenarios.

As an example, an individual states a set of "privacy preferences" which describe what the individual does or does not want to allow the calendaring service to do with the personal information. The calendaring service uses a set of "privacy practice rules", to make decisions about use and disclosure of personally information. The calendar service makes the decision by combines the privacy practice rules with the privacy preferences to determine whether a proposed use or disclosure is permissible.

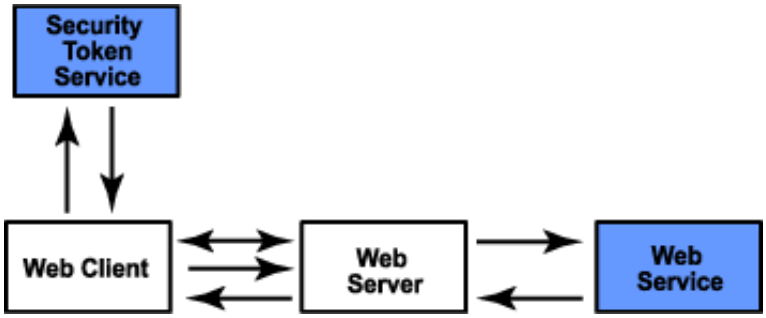
Figure 12. Privacy statements in service policies



Web Clients

[8] Consider an example where we have a Web client communicating with a middle-tier Web application, which, in turn, is talking (securely) to a Web service in another domain. The middle-tier Web application, which is Web services-aware, wants to obtain a security token for the Web client.

Figure 13. A Web client communicating through a middle-tier application to a service



Furthermore, this scenario assumes that the security token enables the middle-tier application to act on the client's behalf when it talks to the service.

To enable this scenario, the Web client's browser accesses the middle-tier application and is redirected to an associated identity service. Once authenticated (for example using a HTML form and https), the request is redirected back to the middle-tier application. The identity service provides a security token (asserting the identity and the delegations) to the original middle-tier application Web server (for example using a query string sent via HTTPS). The Web server can now use these security tokens and issue requests from its own identity to the Web service. The Web service processes the requests, and returns the results to the Web server, where the results are formatted and returned to the browser.

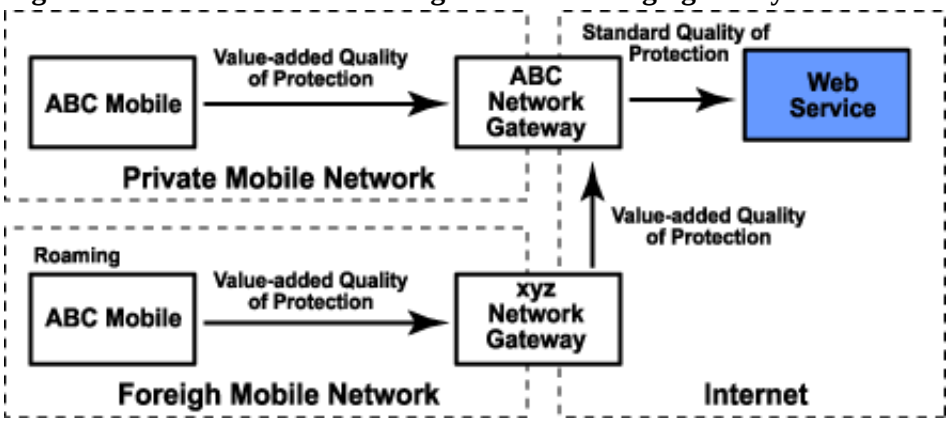
Mobile Clients

The specifications described in this document above provide substantial flexibility in addressing the design challenges that are unique to mobile security. The flexibility of the Web services approach enables support for multiple cryptographic technologies providing both strong and performant cryptographic protection on devices with limited computational and storage capabilities. Similarly it enables network operators to provide security proxies, such as network gateways,to act on the mobile clients' behalf.

Following is an example combining both these ideas. When a network operator supports mobile clients (using these Web service security specifications) they can configure those clients to send requests via the network operator's gateway.In this scenario the gateway is a SOAP intermediary that actively participates in the overall message flow; specifically, the network operator is providing a value-add encryption algorithm designed for mobile devices.The gateway can augment or change the security tokens and quality of protection of the message. Note that the flexibility inherent in this Web services security model allows this solution even when the device is roaming on a foreign network.

This is illustrated in the example below:

Figure 14. Mobile clients accessing a service through gateways



Enabling Federation

[9] The Web services Security model is designed to support federation. Here is a simple example of identity federation:

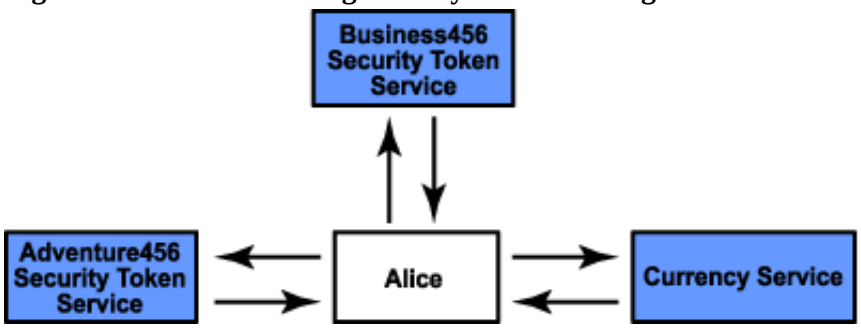
Alice at Adventure456 wants to use the Currency Web service at Business456. The Currency service will only process requests with a security token issued by Business456. Alice only has a security token with identity claims (i.e. an identity security token) issued by Adventure456.

In this scenario, Alice will only be able to access the Currency service if Business456 is willing to accept security federation with Adventure456.

The following subsections describe several approaches to security federation.

Federation Using Security Token Exchange
In this approach, the Currency service's policy states that it only accepts security tokens issued by Business456. Because the policy indicates where to get the required security token, Alice presents (and proves) her Adventure456 security token to the Business456 security token service and receives a Business456 security token. She then presents (and proves) this security token in requests to the Currency service. This is illustrated in the diagram below:

Figure 15. Federation using security token exchange



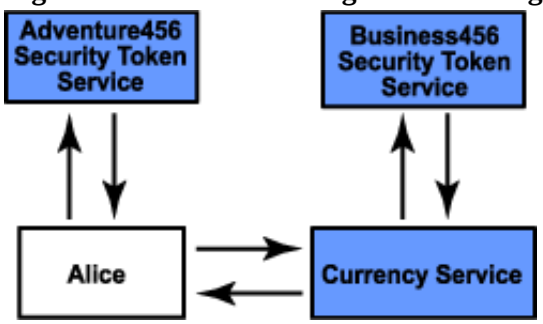
In this approach the Business456 security token service was configured to accept security tokens with identity claims issued by Adventure456.

It should be noted that this example is very similar to the example in the Enforcing Business Policy scenario. This demonstrates the flexibility of the web service security model.

Federation Using Trust Chaining
In this approach, the Currency service will accept a request with any security token, but it will not process the request unless it can obtain a Business456 security token based on the provided security token (and proof).

To do this, the Currency service forwards the original request to the Business456 security token service which evaluates the initial security token. If valid, it endorses the request and may include a Business456 security token for Alice to use on subsequent requests. This is illustrated in the figure below.

Figure 16. Federation using trust chaining

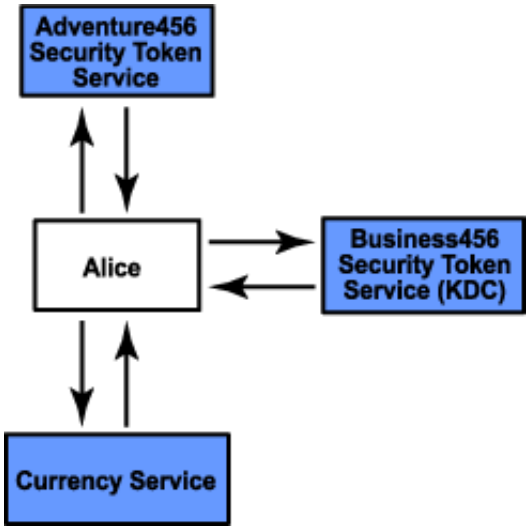


In this approach the Business456 security token service was configured to accept security tokens with identity claims issued by Adventure456

Federation Using Security Token Exchange (PKI -> Kerberos)
In this approach Adventure456 has issued Alice a public key security token and the Currency service's policy indicates that it only accepts Kerberos security tokens from its KDC.

At the direction of the Currency service's policy, Alice presents (and proves) her public key security token to Business456's security token service. The security token service encapsulates Business456's KDC. As a result, it is able to validate Alice's public key security token and issues a Kerberos security token for the Currency service. This is illustrated in the figure below:

Figure 17. Federation using security token exchange (PKI -> Kerberos)



In this approach the Business456 security token service was configured to accept public key security tokens with identity claims issued by Adventure456.

Federation Using Security Token Exchange (Kerberos -> Security Token Service)
In this approach Adventure456 has issued Alice a Kerberos security token and the Currency service's policy indicates that it only accepts security tokens issued by its own security token service.

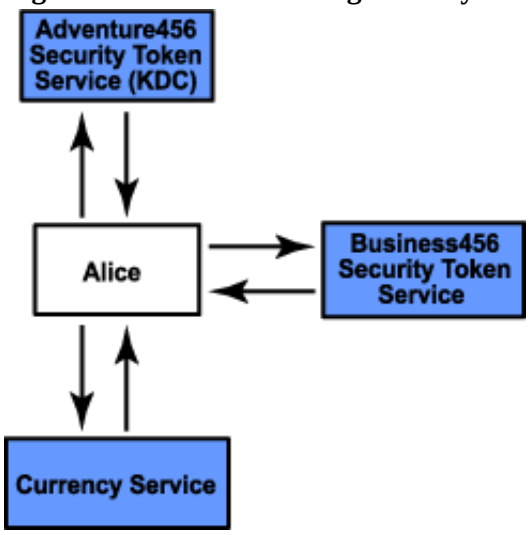
Here we assume the administrators at Adventure456 and Business456 have exchanged public key certificates in order to federate security. We further assume that Alice only supports symmetric key technology.

Based on the Currency Web service policy, Alice needs to acquire a security token that can be used to access the security token service at Business456. Because Alice is using symmetric key security tokens, Alice first contacts her security token service to acquire a security token that is intended for the Business456 security token service. Note that this security token will contain a symmetric key, Sab, encoded with the public key of the Business456 security token service.

Using the security token intended for the Business456 security token service, Alice requests a security token for the Currency service. The Business456 security token service provides Alice with a symmetric key, Sac, and a security token for the Currency service.

Using the security token intended for the Currency service and the associated symmetric key, Alice makes requests to the Currency service.

Figure 18. Federation Using Security Token Exchange (Kerberos -> Security Token Service)



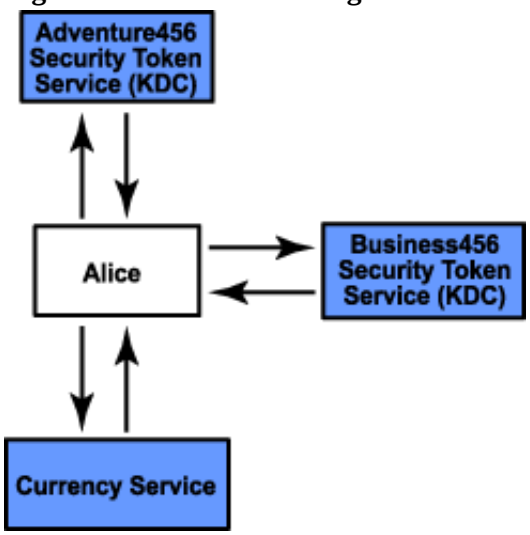
Federation Using Credential Exchange (Kerberos -> Kerberos)
In this approach Adventure456 has issued Alice a Kerberos security token and the Currency service's

policy indicates that it only accepts Kerberos security tokens issued by its own security token service that encapsulates its KDC.

Here we assume the administrators at Adventure456 and Business456 do not want to establish cross realm transitive trust but are willing to exchange public key certificates in order to federate security. Further we assume that both Alice and the Currency service only support symmetric key technology.

As in the previous example, the security token services have the ability to provide symmetric key security tokens to the services within their trust domain. As a consequence, the approach described above works in this scenario.

Figure 19. Federation Using Credential Exchange (Kerberos -> Kerberos)

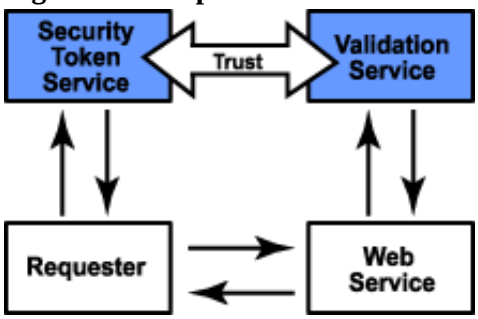


Validation Service

[10] This Web services Security model also supports scenarios in which the requester *outsources* the processing of the message and security token validation. It should be noted that misuse of such a service can cause scalability issues. That is, depending on the implementation it may be cheaper -- or more expensive -- for the service to perform the validation service. The Web services security model supports either approach, thereby enabling this scenario.

In this scenario we have separated the validation service from the security token service. In other scenarios they could be combined or have a direct trust relationship (therefore not requiring WS-Federation).

Figure 20. A separate validation service from the security token service



In this scenario, the requester obtains a security token from the Security Token Service. It then sends a message to the Web service and includes proof-of-possession (e.g. a signature). The Web service sends the signature block, the security token, and its computation of the digest that was signed to the validation service. The validation service, which is trusted by the Web service, then issues a valid/invalid decision.

It should be noted that the validation service could indicate its decision by issuing a security token on behalf of the requester that asserts the appropriate claims.

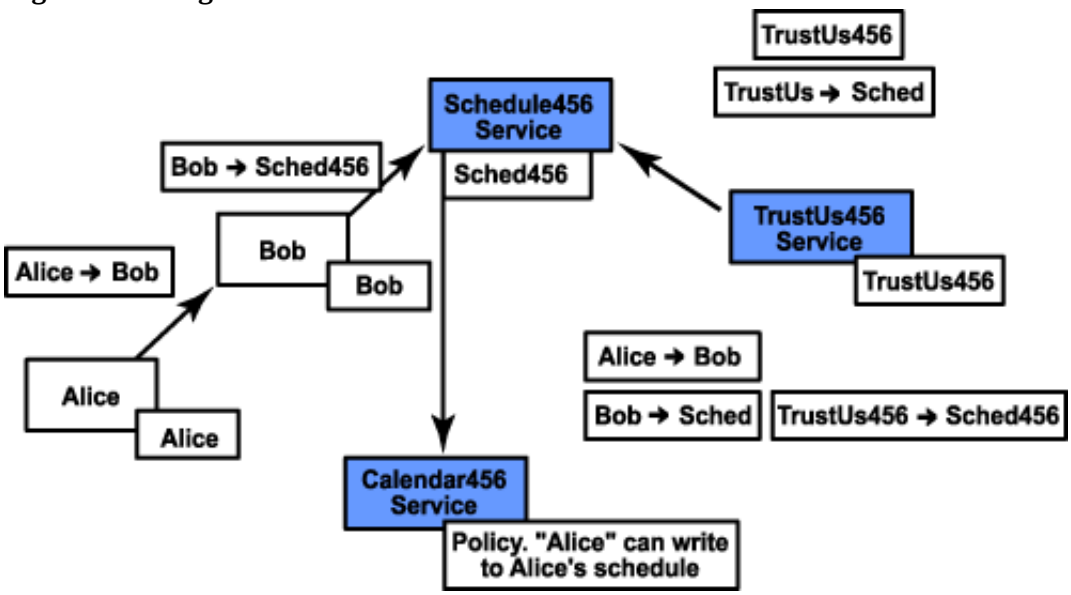
Supporting Delegation

[11] Web services security supports delegation. Here is a sophisticated delegation scenario designed to show the flexibility of the approach: Alice uses calendar456 to manage her schedule. She wants to allow Bob to

set up a meeting with her on Tuesday. However, Bob does not do the scheduling directly. Instead Bob uses the service, schedule456, to set up meetings that need to be put on Alice's calendar.

Alice doesn't know how Bob will schedule the meeting, but she wants to limit the set of services that can see her calendar.

Figure 21. Delegation of trust



Alice provides Bob with a security token that enables Bob to schedule meetings on her calendar. This security token contains claims that restrict Bob's ability to schedule meetings to Tuesday. Furthermore, this security token enables Bob to issue subsequent security tokens so long as the subject(s) can prove they have a privacy certification from TrustUs456, a third party service.

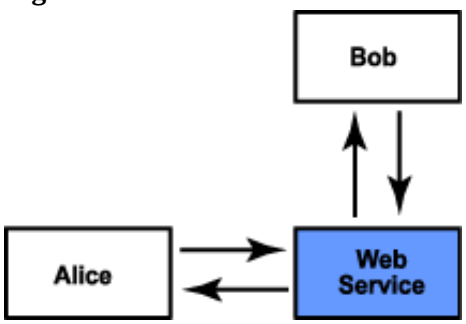
Since the service schedule456 has been audited by TrustUs456, they have a security token that asserts their privacy certification.

When the schedule service accesses Alice's calendar at calendar456, it can prove proof-of-possession of its privacy certification and its claim to access and schedule a meeting based on the security token(s) from Alice, through Bob, to itself.

Access Control

While working together, Alice and Bob find that they are frequently scheduling meetings with each other and develop a level of trust. Consequently, Alice wants to allow Bob to schedule meetings without having to delegate to him every time. She could increase the expiration of the delegation security token, but she will need to re-issue them and this is problematic if she wants to rescind Bob's ability to schedule meetings.

Figure 22. Access Control



Alice communicates with her calendar service (authenticates herself) and obtains the authorization list. She updates the authorization list to allow Bob to see her free/busy data and schedule meetings and submits it to the service. Now when Bob accesses her calendar service for these operations, he doesn't need a delegation security token from Alice.

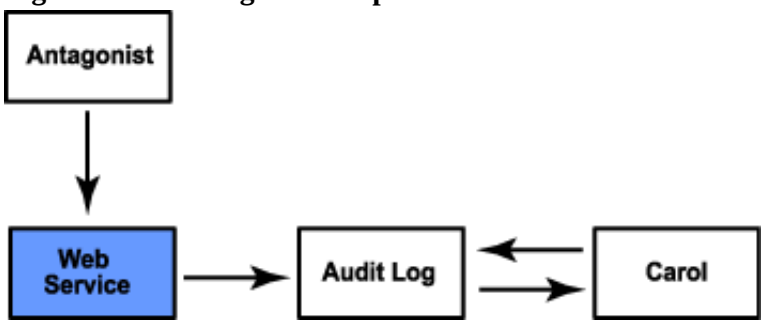
Auditing

[12] In the delegation scenario above, it is possible that an antagonist might try to schedule a meeting without a delegation security token or with an expired security token. In such cases the request will fail because the antagonist cannot prove the required claims.

In order to track this type of activity, the service may provide auditing features. That is, when a security-related event such as authentication or an unproven claim or a bad signature occurs, it is logged. An administrator can securely access the log to review security-related events and manage the log.

For example, antagonists may try to imitate Bob. Using a monitor/management tool, the security administrator, Carol, reviews the audit log and sees that Alice's calendar has had a number of security failures. In reviewing the data she sees that sometimes Bob's request fail because his signature does not match the message or messages are old (replayed). As a result Carol collects the audit records for use in trying to track down the antagonists.

Figure 23. Auditing service operations



Summary

As Web services are applied more broadly, as application topologies continue to evolve to support intermediaries such as firewalls, load balancers, and messaging hubs, and as awareness of the threats organizations face becomes more well understood, the need for additional security specifications for Web services grows clear. In this document, we propose an integrated Web services security model and a set of specifications for realizing that model. These new specifications, by extending and leveraging (rather than replacing) existing security technology and assets, will enable customers and organizations to more rapidly develop secure, interoperable Web services.

IBM and Microsoft believe that this is the first step in defining a comprehensive Web services security strategy. It reflects the challenges and solutions we have identified thus far. As we continue to work together with customers, partners and standards organizations to secure Web services, we expect that there will be additional ideas and specifications needed to make the strategy complete.

Contributors

This document was jointly authored by IBM and Microsoft.

Key contributors include (alphabetically): Giovanni Della-Libera, Microsoft; Brendan Dixon, Microsoft; Joel Farrell, IBM; Praerit Garg, Microsoft; Maryann Hondo, IBM; Chris Kaler, Microsoft; Butler Lampson, Microsoft; Kelvin Lawrence, IBM; Andrew Layman, Microsoft; Paul Leach, Microsoft; John Manferdelli, Microsoft; Hiroshi Maruyama, IBM; Anthony Nadalin, IBM; Nataraj Nagaratnam, IBM; Rick Rashid, Microsoft; John Shewchuk, Microsoft; Dan Simon, Microsoft; Ajamu Wesley, IBM

Resources

Footnotes

1. Uses WS-Security -- May use WS-Policy
2. Uses WS-Security and WS-Trust -- May use WS-Policy
3. Uses WS-Security and WS-Trust -- May use WS-Policy
4. Uses WS-Security and WS-Trust -- May use WS-Policy
5. Uses WS-Security, WS-Trust, WS-Policy, and WS-SecureConversation -- May use WS-Federation
6. Uses WS-Security, WS-Trust, WS-Policy, and WS-SecureConversation -- May use WS-Federation and WS-Authorization

7. Uses WS-Security, WS-Policy, and WS-Privacy
8. Uses WS-Security and a security token format that supports delegation
9. Uses WS-Security, WS-Trust, WS-Policy and WS-Federation -- May use WS-SecureConversation
10. Uses WS-Security, WS-Trust, and WS-Federation -- May use WS-Policy and WS-SecureConversation
11. Uses WS-Security, WS-Trust, WS-SecureConversation, WS-Federation, WS-Privacy and a security token that allows delegation -- May use WS-Policy
12. Uses WS-Security, Secure Communication, and WS-Trust -- May use WS-Federation

References

- [Kerberos] - J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)," RFC 1510, September 1993.
- [SOAP] - W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.
- [URI] - T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.
- [XML-C14N] - W3C Recommendation, "Canonical XML Version 1.0," 15 March 2001.
- [XML-Encrypt] - W3C Working Draft, "XML Encryption Syntax and Processing," 04 March 2002.
- [XML-ns] - W3C Recommendation, "Namespaces in XML," 14 January 1999.
- [XML-Schema1] - W3C Recommendation, "XML Schema Part 1: Structures,"2 May 2001.
- [XML-Schema2] - W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001.
- [XML Signature] - W3C Proposed Recommendation, "XML Signature Syntax and Processing," 20 August 2001.
- [WS-Routing] - H. Nielsen, S. Thatte, "Web Services Routing Protocol", Microsoft, October 2001
- [X509] - S. Santesson, et al,"Internet X.509 Public Key Infrastructure Qualified Certificates Profile," March 2000