Network Security Guidelines
And Recommendations

Submitted February 28, 2003

Revised April 16, 2003

WATP No. 0601

Task Order No.: T0002AJM038
Contract No.: GS00T99ALD0203

Environmental Information
eXchange
Network

Abstract

This document describes security risks associated with the Environmental Information Exchange Network (Exchange Network). It reviews security models and encryption technologies. It then presents general guidelines and minimum recommendations for securing network nodes.

Table of Contents

Table of Figures

1 Introduction and Terminology

1.1 Introduction

This document focuses exclusively on Web service security issues. It describes Web service vulnerabilities and proposes technical strategies and a roadmap whereby the network group can produce and implement a standards-based architecture that is comprehensive, yet flexible enough to meet the security needs of conducting network exchanges.

As the node group begins to exchange data using Web services, the benefits of having a loosely coupled, language-neutral, platform-independent way of linking applications across local and federal governments, across organizations, and across the Internet are clearly critical to system functionality. At the same time, Web services also raise demanding security challenges that have not been previously encountered.

The Environmental Information Exchange Network (Exchange Network) consists of many network nodes and possibly tens of thousands of users. To secure such a widely distributed network across different network boundaries on various platforms is a very complex task.

1.2 Acronyms and Terminology

Term	Definition/Clarification
ASN.1	Abstract Syntax Notation 1
ACL	Access Control List
CA	Certificate Authority
CAS	Central Authentication Service
CDX Node	The Central Data Exchange Node provided by EPA
CKMS	Central Key Management Services
CRL	Certificate Revocation List
DMZ	Demilitarized Zone
DNS	Domain Name Service
DoA	Detection of Abnormalities
DoM	Detection of Misuse
DoS	Denial of Service
EPA	Environmental Protection Agency
Exchange Network	Environmental Information Exchange Network
ICMP	Internet Control Messages Protocol
IDS	Intrusion Detection System
IRT	Incident Response Team

Term	Definition/Clarification
ITU-T	International Telecommunications Union – Telecommunication Standardization Sector
KISS	Key Information Service Specification
KRSS	Key Registration Service Specification
MAC	Message Authentication Code
MITM	Man in the Middle
NAAS	Network Authentication and Authorization Services. This is a set of centralized security services shared by all network nodes.
OASIS	Organization for the Advancement of Structured Information Standards
OSI	Open Systems Interconnect
PKI	Public Key Infrastructure
RBAC	Role-Based Access Control
SAML	Security Assertion Markup Language
SSO	Single Sign-on
SOAP	Simple Object Access Protocol[1]
SSL	Secure Sockets Layer
W3C	World Wide Web Consortium
WSE	Web Service Enhancements
XACML	Extensible Access Control Markup Language
WSTK	Web Service Toolkit
XKMS	XML Key Management Services

2 Recommendations

The following security recommendations should be deployed on a network node. Detailed background information supporting the recommendations for each security requirement has been provided in subsequent sections of this document.

- **Authenticate Users:** All nodes must ensure users are authenticated before performing requested tasks. Authentication can be done locally or through Network Authentication and Authorization Services (NAAS).

- **Validate Security Token:** All Web methods, except NodePing, have a securityToken parameter. It MUST be validated before performing any other operations.

- **Restrict IP Addresses:** If a node has a small list of legitimate users, it is prudent to give access to only the valid IP addresses, or a range of valid IP addresses. Requests from all other machines should be rejected.

- **Use Secure Sockets Layer (SSL) Protocol:** This is perhaps the most secure way to protect a node. We strongly recommend all exchanges be conducted over 128-bit SSL. Also, disabling of port 80 in production node, so that the only way to reach the service is through SSL, is recommended.

- **Limit the Maximum Message Size:** This prevents hackers from sending large amounts of useless data to network nodes. Most of the Web server implementations offer configurable settings for the size parameter.

- **Validate Parameters Early:** When a request is received, the first step is to validate the submitted data. This includes checking data types, special characters, and parameter values. Parameter validation can ward off many SQL parameter attacks from malicious users.

- **Use Personalized Folder:** For network nodes that support the Download method, we strongly recommend a personalized folder for each user. This prevents unauthorized document access.

- **Use NAAS:** Highly valuable to network nodes, these services are continuously being enhanced to provide higher levels of security. Strong encryption and complex algorithms are combined to offer improved authentication of network users. Use of NAAS can enhance node security and reduce cost.

- **Host Services in a Demilitarized Zone (DMZ):** Network nodes should be hosted in a DMZ, which isolates node services from internal networks through a firewall. The firewall should also use IP sequence number randomization, and stealth on the ports used to contact the node (i.e., Port does not respond to TCP/IP ping. Consult your network administrator for assistance.)

3 Web Service Vulnerability

3.1 Major Threats

The following is a list of typical threats to information security in distributed Web services network.

- **Unauthorized Access to Web Services**: This class of threat occurs when a user attempts to invoke a method in a network node for which the user has no authorization. This often happens when a user only has authorization to node A, but penetrates into node B.

- **Unauthorized Information Flow**: This threat occurs when the flow of information between two nodes is not suitably restricted so as not to violate a prescribed set of flow policies or a partner agreement. For instance, a user may submit documents to the CDX node using the Send interface, but the user may not be authorized to download documents from the CDX node. Unauthorized document flows from CDX node to a user machine are considered dangerous.

- **Vandalism and Sabotage**: This class of threat is aimed at disrupting normal operation of node services by destroying software or hardware on the node server.

- **Theft and Fraud**: This threat is similar to an unauthorized information flow. It is however, aimed at stealing other valuable information from a network node. For example, user registration information, database tables, secret keys (private keys).

- **Violation of Data Integrity**: This threat occurs when a user sends data or documents to a network node, but the data is tampered with while en route to its destination. The network node receives either partial or falsified information as a result.

3.2 Attacks

3.2.1 Forms of Attacks

Web service attacks come in two (2) forms: one against data, the other against control systems. The first attempts to steal or corrupt data and deny services. The majority of Internet and other computer attacks fall into this category.

Control system attacks attempt to disable or take power over operations used to maintain physical infrastructure.

3.2.2 Replay Attack

A replay attack occurs when an attacker intercepts a message with a valid security token, and sends it again and again to the network node. This attack doesn't require any knowledge of the message contents; the Simple Object Access Protocol (SOAP)

message may even be signed or encrypted. The goals of such attacks are to gain access to the node services and/or to disrupt normal operations.

The Authenticate message, defined in the Node Functional Specification, has the highest risk for replay attacks. The attacker may intercept a request message or response message and replay it to gain access to a network node at a later time.

Replay attacks can be prevented if there is a timestamp in the message and there is an expiration time imposed by the receiver. A replay of such a message at a later time will be invalid and rejected by network nodes. The Network Authentication and Authorization Services use a timestamp in the security token and validate token life span when received.

Using a secure communication channel, such as SSL, can also reduce the risk of replay attack because it would be difficult intercept an encrypted message and to replay it entirely.[2]

3.2.3 Eavesdropping

Communications between two network nodes are carried typically through public Internet networks. This makes eavesdropping possible. Information remains intact during eavesdropping. Hackers usually want to intercept information using eavesdropping techniques. Once intercepted, hackers can take the data offline to analyze and decrypt.

It has been proven that even low SSL encryption, such as 54-bit encryption, can be cracked using fast computer systems. Therefore, to prevent eavesdropping, symmetric encryption with 128-bit key and public key encryption with 1024-bit key is required.

3.2.4 Dictionary Attack

Dictionary attack is a common form of attack against password-based authentication schemes. It occurs when an attacker captures the messages exchanged during a legitimate run of the protocol and uses that information to verify a series of guessed passwords taken from a precompiled dictionary of common passwords. This attack works because users often choose simple, easy-to-remember passwords, which, invariably, are also easy to guess.

In the network node environment, eavesdropping on a TCP/IP network cannot be carried out easily because the authentication information is sent through SSL. The attacker, however, can perform attacks directly on a network node by calling the Authenticate method repeatedly using a known user name and a set of passwords from a dictionary.

3.2.5 Message Insertion and Deletion

This attack occurs when a hacker intercepts a network message and makes changes to the message before relaying it to the recipient. The integrity of the message is compromised.

3.2.6 Man In the Middle (MITM)

The MITM attack is subtler than most. The hacker captures the request message and stops it from going any further. He generates his own request message with suitable modifications and sends it along to the Web service. When the response message arrives, he transmits a modified response to the original sender, who is not aware that there is anything wrong.

MITM is a common form of attack because a request message travels through many devices such as routers, gateways and bridges before reaching its final destination. In addition, SOAP messages may be routed through many of those same devices.

MITM attacks can be reduced through SSL encryption. It is difficult for a middleman to relay SSL encrypted messages.

MITM attacks can be prevented using digital signatures. A signed message from the sender guarantees the integrity of its contents. If it is possible to tamper with the digitally signed message without being detected, it is extremely difficult. We will discuss digital signature, particularly XML signature technology, in the subsequent sections.

3.2.7 Denial of Service (DoS)

When the DoS attack is attempted, the attacker tries to disable services and prevent legitimate users from using the services. Forms of DoS include:

1. Flooding a network node with invalid requests, for instance, TCP SYN flooding
2. Disrupting connections between two machines
3. Preventing a particular user from accessing the service
4. Consuming close to 100% of resources on the server

DoS is one of the most common attacks on the Internet; but, unfortunately, there is no complete solution to prevent it. DoS items 1 and 2 above are sometimes referred to as network-level attacks. A perpetrator sends a large amount of ICMP echo (ping) traffic at IP broadcast addresses, all of it having a spoofed source address of a victim. If the routing device delivering traffic to those broadcast addresses performs the IP broadcast, most hosts on that IP network will take the ICMP echo request and reply to it with an echo thereby multiplying the traffic by the number of hosts responding. On a multi-access broadcast network, potentially hundreds of machines could reply to each packet.

The network-level attacks are dangerous, but they are primarily the responsibility of network administrators rather than network builders and operators.

There is an increasing number of DoS attacks aimed at Web services on the application level. The attacker, in such cases, tries to exploit "bugs" in the implementation of Web services and to disable services by consuming 100% of the system resources, such as CPU time. For instance, a hacker may try to cause a buffer-over-run and keep the CPU busy in an infinite loop. Another example of a DoS attack occurs when a hacker sends a very large file (i.e., 300 Gigabytes) to a network node in an attempt to jam the network completely and prevent others from access.

In general, to prevent DoS attacks at the application level, a network node should:

- Limit the number of resource intensive operations unless the request comes from an authenticated source. The securityToken should always be validated prior to any other operations,

- Patch security holes such as buffer-over-run and take care of error/exception handling

- Limit the maximum bytes a client can send or retrieve.

The last one is especially important for preventing DoS attacks using the Submit method where a hacker submits a huge file to jam network services.

3.2.8 IP Spoofing

Spoofing is the creation of TCP/IP packets using a valid user's IP address. Routers use the "destination IP" address in order to forward packets through the Internet, but ignore the "source IP" address. The destination machine only uses the address when it responds back to the source.

IP spoofing is a common technique for disabling a network service. The sender is not interested in receiving a response message in this case, but intends to flood the victim with useless network requests.

3.2.9 SQL Parameter Attack

This kind of attack has become very common in recent years, and has caused more serious damage compared with other attacks. A SQL parameter attack uses special characters to cause Web services to return restricted information on the server. For example, a SQL query statement to return user information may be constructed as

sqlQuery = Select * from users where (UserName='" & UserInput & ")"

If a malicious user entered

Joe') or not (UserName='0

as the value of UserInput, then the service will return all user records which causes a security breach.

The SQL Parameter Attack is a primary concern of network node security because most of the data exchanges rely on SQL query operations implemented using stored procedures. Almost all such service requests have parameters that could potentially be exploited.

Network nodes should validate all parameters before executing an SQL statement. This is especially important for the Query and Solicit methods, where input parameters are passed as an array of strings.

4   Security Requirements

There is no such thing as perfect security. However, a network node is considered secure or strong if it meets the following general requirements:

- **Authentication:** The Exchange Network is a protected resource. All nodes must ensure that messages actually originated at the source from which they claim to have originated. Authentication should include user authentication, machine authentication, and message authentication.
- **Authorization:** Requesters should only be allowed to access node services they are authorized to access. Note that authorization depends on authenticated user identity in order to grant access based on security rights.
- **Confidentiality:** For some data exchanges, it is important that messages are not visible to anyone except the two parties involved. This means traffic will need to be encrypted so that machines in the middle cannot read the messages. In addition, documents, in part or in whole, may contain classified information or highly sensitive data. Even the network node, if it is not the ultimate receiver, may not be allowed to read the message.
- **Message Integrity:** It must be guaranteed that messages between network nodes have not been tampered with during transit.
- **Non-repudiation:** Documents exchanged may be required, by either state or federal laws, to contain a digital signature to guarantee that the sender of the message cannot deny their identity as the sender.

5   Web Service Security Model

A security model is a representation of the security policy and deals with the security properties (such as rules and constraints) of the system. In an information system, the security model is abstract and generic and should only contain information pertinent to the security aspects of the system.

This section discusses some of the common security models. In a distributed system such as the Exchange Network, implementation of a security system may require a combination of multiple security schemes.[3]

5.1   Basic Authentication

Basic authentication uses username and password for user identity verification, and grants user access to all resources of a Web service if successful. Basic authentication is very simple to implement, and it is currently the most popular security model used for websites and Web services. The model, however, is very weak and subject to many attacks.

Basic Authentication over HTTP is very problematic and must not be used by all network nodes.

5.2   Transport Security

Transport Security uses existing technologies such as secure sockets (SSL/TLS) that can provide simple point-to-point integrity and confidentiality for network messages. SSL provides a secure communication channel between network nodes, but it is insufficient because it only provides confidentiality and integrity protection at the network packet level, not on the XML message level. Classified information in an XML document can be compromised after being decrypted at the receiver end.

5.3   Multi-level Security

A multi-level security scheme has been used in the military for quite some time. Information is classified into multiple categories: Unclassified, Confidential, Secret and Top Secret. Access to a higher-level security group requires additional credentials. A multi-level security model has real application in the Exchange Network. Documents exchanged through the network can be classified in a manner similar to the military scheme, using the four (4) levels of Public Access, SSL with Client Authentication, SSL with dual authentication, and digital signatures that are cited in the Network Exchange Protocol document. A higher-level security document may not only be transferred through a secure channel, but also encrypted using high strength cipher, so that only the ultimate receiver can see the content.

5.4   Chinese Wall Model

The Chinese Wall Model is an access control policy proposed by Brewer and Nash in 1989.[4] This popular model was proposed for information flow in a commercial sector. It is defined as follows: All organization information is categorized into mutually disjoint conflict of interest classes. The Chinese wall policy states that information flows from

one origination to another that cause conflict of interest for individuals should be prevented.[5]

The Chinese Wall model does not apply to the Exchange Network because all nodes within the network are considered partners. The model would apply when there are multiple exchange networks and information flows from one network to another that need to be tightly controlled.

5.5   Public Key Infrastructure (PKI)

PKI uses a pair of asymmetric digital keys to encrypt transmitted data. The PKI model also involves certificate authorities issuing certificates with public asymmetric keys and authorities that 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.

While there are some success stories about PKI, there are risks involved when using PKI.[6] In addition, there are several disadvantages of using PKI in a distributed environment such as the Exchange Network:

1. Lack of standard interfaces: There are no standard interfaces for PKI. Each implementation uses proprietary technology, which causes wide interoperability issues.
2. Management complexity: Managing certificates distributed across state governments and the federal government is a towering task. It is even more complicated if the certificates are issued by multiple certificate authorities without trust relationships.
3. High cost: Acquisition of a total PKI solution is not cost effective for this application.

6   Web Service Security Standards

Although Web services are powerful technologies to build cross network applications, they also introduce security risks. Security is the number one concern for deploying Web services in organizations and enterprises. To respond to these concerns, communities and standards bodies are developing standards and technologies to secure Web service environments.

Work on the Web service security specifications within the World Wide Web Consortium (W3C) and other major players is still in progress, but the foundation for securing Web services is in place at this time. Figure 1 shows some the key security specifications that are either available now or will be available soon.[7]
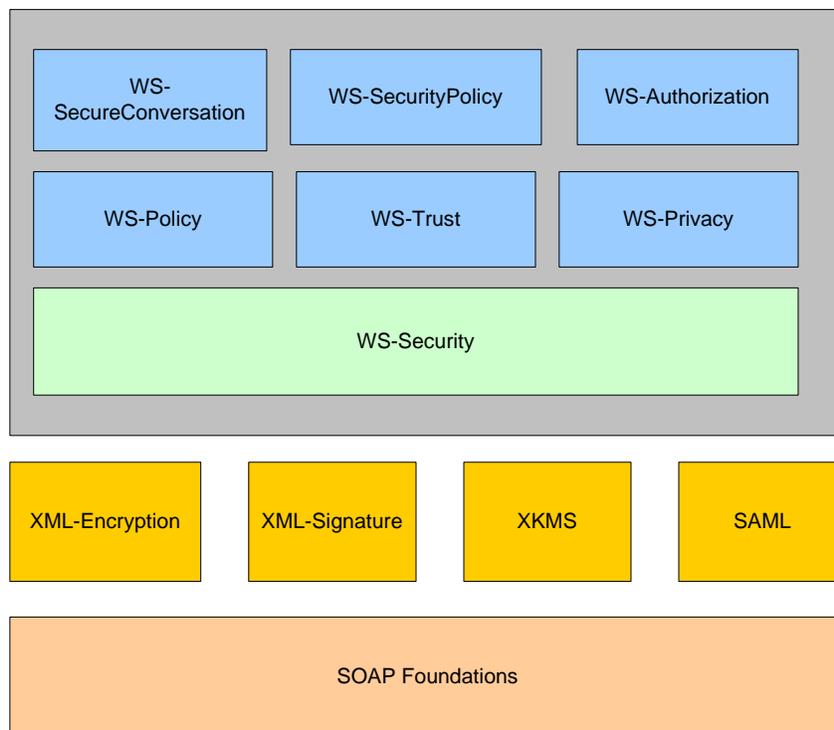
Figure 1: Web Service Security Specifications

In this section, some of the specifications that are related to the Exchange Network are briefly described. They will be further elaborated in subsequent sections where we discuss how to use the specifications to secure network nodes.

6.1   XML Encryption

XML Encryption, a W3C recommendation since December 10, 2001 defines rules to encrypt data in an arbitrary format and represent it in XML format. It also defines a standard process for decrypting data. XML Encryption is a key foundation technology and a crucial component of the Web services security stack.[8]

6.2 XML Signature

XML Digital Signature is a standard for securely verifying the origins of messages. The XML Digital Signature specification allows XML documents to be signed in a standard way, with a variety of different digital signature algorithms. Digital signatures can be used for validation of messages and for non-repudiation.[9]

6.3 WS-Security

The OASIS standards body consortium's WS-Security specification defines the format and information required for a secure SOAP exchange. Specifically the specification addresses four (4) functional areas; digital signature format, security token format, trust domain, and encryption method. The WS-Security standard further classifies these functional areas into three (3) mechanisms:

1. Secure token transmission

2. Message integrity

3. Message confidentiality

WS-Security was designed to be "Standards" friendly, to existing standards, and each of these mechanisms supports a variety of existing implementation techniques. These three mechanisms of the WS-Security specification can be used independently or in combination. Note that WS-Security standard does not define how functional areas are created. It is up to the standards for WS-SecureConversation, WS-Trust, WS-Policy and others (see Figure 1 on the prior page) to provide the definition of the functional areas.

The security token mechanism in WS-Security is very flexible. In WS-Security, a security token will have a claim associated with it. OASIS defines a claim as a declaration made by an entity (which can be a name, key, group, etc.). Also security tokens are defined as either unsigned or signed. An example of an unsigned security token is a userid (involved with the claim) and password (the security token). A signed security token is associated with a digital signature or certificate.

For message integrity WS-Security allows digital signatures to be carried in the header of the SOAP message. The signature portion is an extension of the XML SOAP signature standard.

Message confidentiality is also very flexible. Any part of the SOAP block can be encrypted. WS-Security encryption is required to support the XML SOAP encryption standard.

Overall WS-Security can be considered an enhancement to SOAP. Because of this enhancement, WS-Security components should be capable of being plugged into existing node services with minimal changes. Since the security features are likely carried inside SOAP headers, it is the responsibility of the SOAP toolkits to process these security extensions (such as XML signature and security tokens). [10]

6.4 Extensible Access Control Markup Language (XACML)

XACML is standardizing security access control using XML by defining an XML specification (core schema and name space) for expressing authorization rules over the Internet.

As of February 6, 2003, XACML 1.0 has officially become an OASIS standard.[11]

6.5 Security Assertion Markup Language (SAML)

SAML is defining a standard XML syntax for specifying authentication and authorization credentials that can be sent along with SOAP messages. SAML will specify additional SOAP headers to carry assertions among many other requirements. [12]

6.6 XML Key Management Services (XKMS)

Public Key Encryption (PKI) can be complex to deploy. This is somewhat in conflict with SOAP, which is built to have less overhead. Consequently, XKMS is being developed to simplify the deployment of PKI in XML. The W3C standards group has identified several requirements for XKMS:

- It should mask the complexities of PKI and expose only the minimum information needed to use PKI.

- Support PKI status information to facilitate secure exchanges.

- Keep the implementation code small, in order to keep SOAP fast and efficient.

- Support the XML signature specification.

- It does not require the Abstract Syntax Notation 1 (ASN.1) standard. This standard was developed by the International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) and the Open Systems Interconnect (OSI) committees. The standard defines the format of data being transmitted by data communication protocols. The data can be binary oriented in ASN.1, this can make the SOAP message more complex since it is XML human readable.

For XKMS to operate, two (2) protocols are being developed under the standard:

- Key Registration Service Specification (KRSS), it provides the services to allow the key holder to register their public keys.

- Key Information Service Specification (KISS), this provides the key information to the SOAP message and masks/simplifies the PKI processing.

After the keys have been registered by XKMS KRSS, secure exchanges are to be handled by KISS. To further simplify XKMS, KISS has been abstracted into two services; locate and validate. Locate finds the key and validate confirms that it is valid.[13,14]

In summary, XKMS is being created to provide an efficient and effective deployment of PKI in SOAP message exchanges. Since the node network will need to support XML

Encryption and XML Signature, we propose a central XKMS service be deployed. All network nodes would share this central XKMS server. More detail is provided in Section 9.6.

6.7   WS-Policy

WS-Policy describes how senders and receivers can specify their requirements and capabilities, including privacy attributes, encoding formats, security token requirements, and supported algorithms. WS-Policy defines a generic policy format that can be used to support security policies associated with a set of Web services.

Released on December 18, 2002, it is expected that the specification will not be implemented and widely supported until next year.[15]

6.8   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.

The WS-Authorization specification has not been released as of the date of this document.

7   Web Service Security Elements

7.1   Authentication

Authentication is the process of verifying that a subject, either a user or a machine, is who they claim to be. (i.e., to certify or prove that someone or something is genuine.) The authentication process requires that the subject present evidence, or a credential. The credential is then checked or verified against an authority.

7.1.1   User Authentication

User authentication is a process of verifying a user's identity.

7.1.1.1 User Name and Password

User name and password is perhaps the most commonly used authentication scheme because it is simple to implement.

The problem with password authentication is that passwords are easily stolen. Many users choose easily remembered passwords that are so obvious that they are easy to figure out.

Password authentication is also subject to dictionary attack (See section 3.2.4).

Since passwords are sent as clear text, all network nodes must use SSL for user authentication.

7.1.1.2 Password Digest

If this method is used, instead of sending password to the service provider, the user sends a mathematically computed digest of the password, also known as the hash value. The digest is generated using a cryptographic algorithm, such as MD5 or SHA1, which creates a unique value derived from the password string. The value, often 128-bits long, is considered a highly reliable fingerprint of the given password string.

The service provider uses the same algorithm to compute a digest and compare it with the received password digest. The user is successfully authenticated if the two digests are equal.

Password digest authentication is considered safer than the password authentication method because passwords are not transferred through the network. It is difficult to reconstruct a password based on its digest.

7.1.1.3 Digital Certificate

Digital certificates (X.509) can also be used to authenticate users. Typically, a certificate is issued to a user, after validating the user identification and other related information, by a trusted third party – the Certificate Authority (CA). The CA signs the certificate so that it cannot be forged, and certifies the information contained in the certificate is genuine.

At the time of authentication, the user presents their certificate to a network node and the network node validates the signature, CA trusts, and the CRL (certificate revocation list) to determine whether or not the login process is successful.

A digital certificate can also be used in transport-level authentication with SSL. A Web server can be configured to perform client authentication. It can check that a client's certificate and public ID are valid and have been issued by a CA listed in the server's list of trusted CAs.

Authentication using an X.509 certificate is much more secure than other authentication methods. However, it requires a PKI and is expensive.

7.1.1.4 Public/Private Key Authentication

This authentication scheme requires the private key holder (user) to first register the public key at a public key depository. The user then signs the authentication message using the private key, and sends the message and signature to the service provider as proof of possession of the private key. Since there is only one private key and the signature can only be validated using the companion public key, the authentication is successful if the signature verification succeeds.

Public key authentication is a viable alternative to a digital certificate if the binding between the key name and key value can be secured.

7.1.2 Machine Authentication

There are situations where a node can only be accessed from identified machines. For instance, a state node may allow only the CDX node to download documents. The process of identifying a machine is called machine authentication.

A popular type of machine authentication is based on an IP address (IP filtering). A node may have a table of valid IP addresses that can connect to the network services; all other requesters are rejected. Many Web servers support this feature, and it can be configured easily. IP filtering devices are also widely available.

Machine authentication can also be accomplished using a shared secret. A node encrypts a string using a shared secret and sends it to another node. The receiver then decrypts the string using the same secret. The machine authentication is successful if the string can be decrypted successfully.

The Central Authentication service (see Section 7.1.4) uses IP addresses to secure the securityToken. A security token issued to one machine cannot be used on another. This prevents security tokens from being stolen.

7.1.3 Message Authentication

Authentication not only can apply to users and machines as discussed above. It can also apply to messages. A process to assure a message's genuineness is called message authentication. Authentication of message contents is of particular importance where there is a risk of either inadvertent or deliberate modification.

A common approach to message authentication is to use Message Authentication Code (MAC). Generation of a MAC is achieved by transforming a message in a way that produces a relatively small datum entirely dependent upon the message content. The MAC is appended to, or incorporated into, the transmitted message. The recipient repeats the MAC generation process over the received message, and verifies that the calculated MAC is identical to that the one supplied with the message. The MAC may also be referred to as a message digest.

A message digest can be understood as the 'fingerprint' of a message. A malicious user could send a bogus message with a correct MAC (a fingerprint of the bogus message). Therefore, a MAC is not particularly usefully until it is tied to the sender – i.e., signed by the sender. We will further discuss XML signature, which is an encryption of MAC using the sender's private key in Section 7.4.2

7.1.4 Central Authentication Service (CAS)

The CAS is a part of the Network Authentication and Authorization Services. Although it is currently based on the basic authentication scheme (password and pin) with SSL, it can be extended to support digest authentication, certificate authentication and SAML.

The central authentication is a set of Web services that provide network-wide security support. This section discusses some of the common application scenarios where the services can be used by network nodes as well as network users.

7.1.4.1 Direct Authentication Model

This model defines the network node as responsible for authenticating the user and it has no direct connection or trust relationship with the central authentication services.

The user, in the scenario depicted in Figure 2, sends an authentication request directly to the node and uses the obtained security token exclusively at the network node.
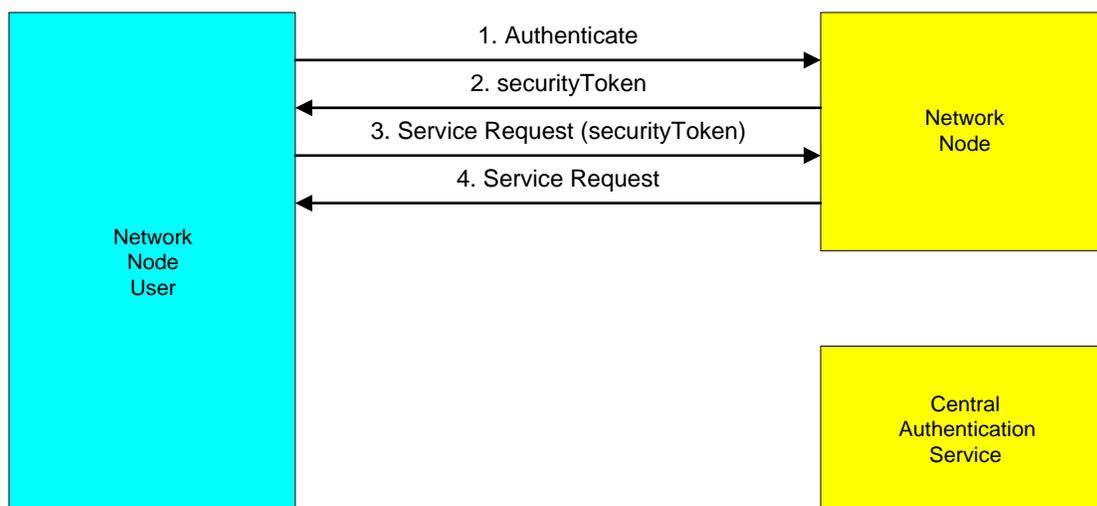

Figure 2: Direct Authentication Model

17

7.1.4.2 Delegated Authentication Model

This simple extension to the direct authentication model is depicted in Figure 3. The user sends an Authenticate message to a node. The node then delegates the request to the NAAS for processing. The user is not aware that there is a NAAS at all.
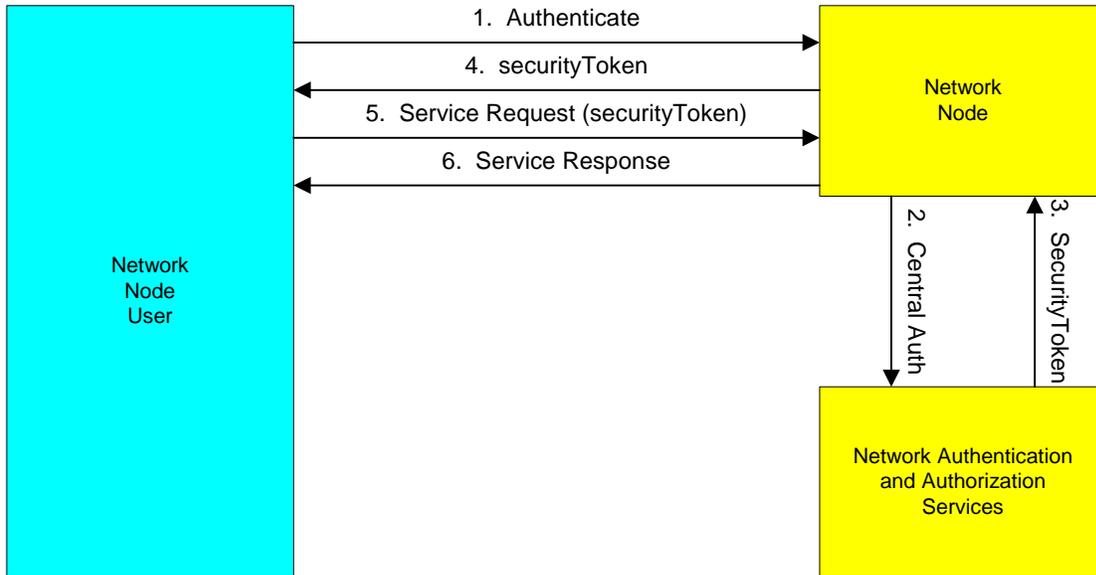
Figure 3: Delegated Authentication Model

7.1.4.3 Direct Trust Model

In this model, a network node trusts decisions made by the CAS, and the CAS authenticates users directly. The trust relationship between the node and NAAS allows assertions such as:

> The central security service said Bob is a good guy (securityToken)

The securityToken issued by the central authentication services is accepted and trusted. Figure 4 shows message exchanges under this scenario.
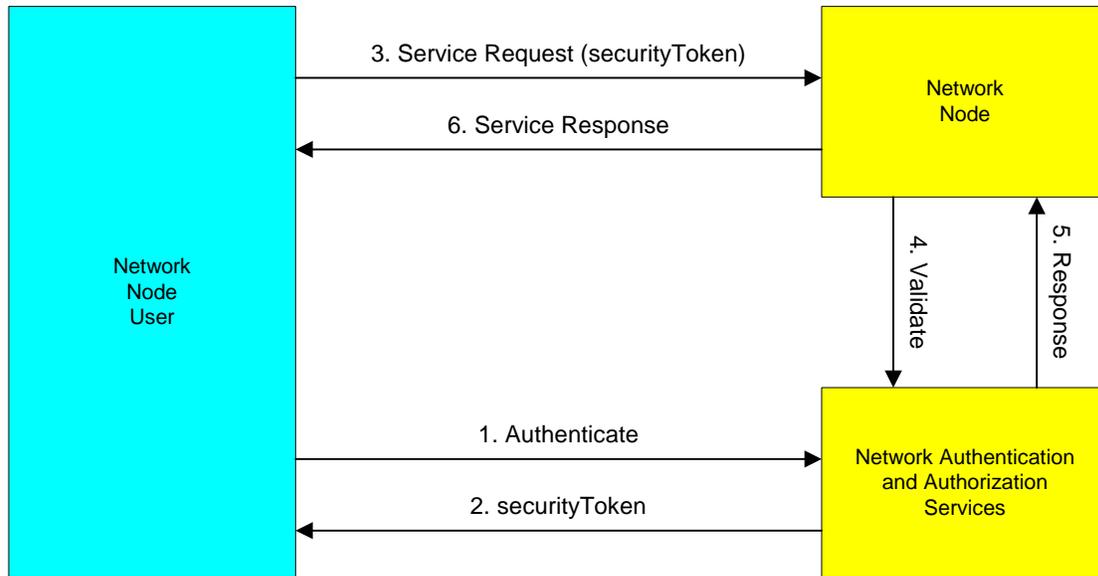
Figure 4: Direct Trust Model

7.1.5 Single Sign-on (SSO)

A securityToken may, ideally, be issued through a central authentication server. The token is recognized and honored by all participating network nodes in a trusted relationship. A central authentication server facilitates SSO. Users need only register or login once in order to access services provided by all network nodes.

SSO can be achieved relatively easily in a centralized authentication environment.

The process of SSO is outlined below:

1. The client sends a name and credential to the authentication server. The server returns a securityToken when successful.

2. The client then invokes a remote method on a network node, using the securityToken.

3. The node sends the securityToken to the authentication server for verification.

4. The authentication server checks the securityToken, and returns either a positive or negative answer.

5. The node processes the request if the securityToken is valid.

7.2 Authorization

Authorization is a process that establishes entitlement of a user. The user or principal, although authenticated, may not be allowed to access certain network services based on a security policy. Given the authenticated user identity (the subject) and the security policy of a network resource (the object), a network node can determine whether or not to grant access. Authorization confirms the service requestor's credentials. It determines if the service requestor is entitled to perform the operation, which can range

from invoking the Web service to executing a certain part of its functionality (executing a stored procedure, for instance, using the Query method).

Authorization determines whether the service provider has granted access to the Web service to the requestor. It answers the question:

Is operation X by principal Y on resource Z permitted?

In order to answer the question, an authorization service typically has a set of policies that bind X, Y and Z together. Decisions are made based on these policies against the requested operation by the principal.

Controlling access to sensitive or confidential resources is a critical requirement of a network node. In the past, this access has been controlled using firewalls. However, this solution becomes totally inadequate for Web services, which allows requestors to penetrate firewalls. What is needed is an infrastructure to easily control users' access to protected applications across the entire Exchange Network.

Authorization must be policy-based, so that large numbers of users can be supported easily. In addition, it must be flexible so that a variety of factors can be part of the decision to grant or deny access to specific resources.[16]

7.2.1 Simple Authorization

In a simple authorization scheme, access rights of a user are determined by checking information about the user identity against some access control information, such as an Access Control List (ACL), which may be a simple matrix similar to the following:

	Authenticate	Submit	Download	Query	Execute	Notify	GetServices
Alice	√	√					
Bob	√	√	√	√	√	√	√
John	√					√	√

In this example, Bob is granted rights to all Web methods at a network node; but Alice can only submit documents.

The ACL can be extended further to service requests a user can perform using the Query method as demonstrated in the following table:

	GetFacilityByName	GetFacilityByLocality	GetNEIDataByMonth	GetNEIDataByYear
Alice	√	√		
Bob	√	√	√	√
John			√	√

It shows that Alice has access to Facility Registry System (FRS) data, but John can only access National Emissions Inventory (NEI) information.

Simple authorization is a means of restricting access to information based on the identity of users and/or membership in certain groups. Access decisions are typically based on the authorizations granted to a user based on the credentials he presented at the time of authentication (user name, password, hardware/software token, etc.).

7.2.2 Role-Based Access Control (RBAC) Authorization

In RBAC, access decisions are based on an individual's roles and responsibilities within the Exchange Network or user base. The process of defining roles is usually based on analyzing the fundamental goals and structure of network management and is usually linked to the security policy.

In role-based authorization, users are categorized into groups based on the roles they play. For instance, user groups may be Administrator, Operator, and User.

Role-based authorization provides network security administrators with the ability to determine who can perform what actions, when, from where, in what order, and in some cases under what relational circumstances.

7.3 Confidentiality

SSL provides only a partial solution to information confidentiality, Using SSL the channel over which two parties communicate can be kept confidential - data is encrypted by the sender and decrypted by the recipient. Documents, or messages, are decrypted at end of the tunnel; they become plain text files that can be viewed by anyone who has access to the machine. In the Exchange Network, however, some data flows require data confidentiality until documents reach the final destination. SSL encryption is thus inadequate for such applications.

7.3.1 Transport Encryption

Confidentiality is assured in most situations where messages are delivered through HTTPS transport. There are several situations, however, where a message may be compromised during transaction if it is not encrypted:

1. Use of transports such as SMTP or FTP.
2. Use of WS-Routing when messages travel over SOAP intermediaries.

It is strongly recommended that messages be encrypted using XML Encryption under such application scenarios.

7.3.2 Shared Secret Encryption

Shared secret encryption is the simplest data encryption technique. In this situation, the sender and the receiver share a secret key, and they use a symmetrical cipher algorithm to encrypt and decrypt data, i.e., the data can be encrypted and decrypted using the same key.

The following Figure 5 shows the process of shared key encryption and decryption:
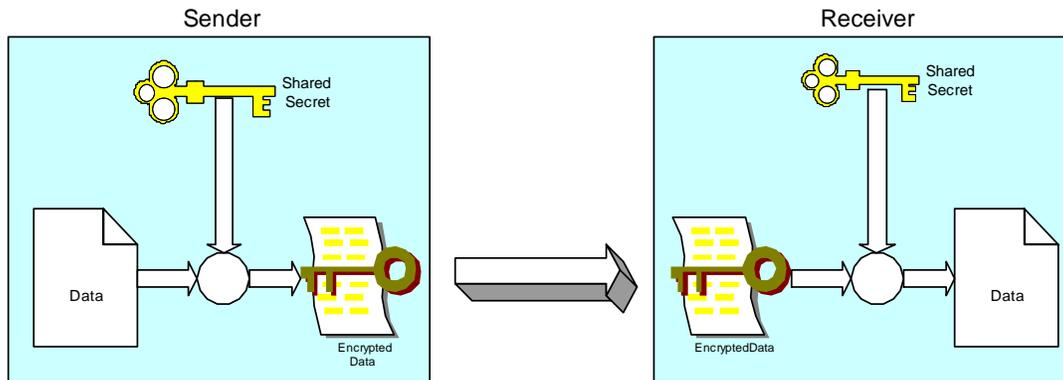
Figure 5: Shared Secret Encryption

Because of simplicity, symmetric encryption and decryption are very efficient. Symmetric encryption has some advantages:

- Works well in environments where keys can be exchanged safely

- Very fast and efficient

- Easy to implement

However, it suffers the following problems:

- Shared keys are no longer secret when shared with a large group of holders. When data exchanges are conducted among many partners, the confidentiality of the keys can no longer be guaranteed.

- Key distribution to a large number of key holders through other channels is subject to interception by adversaries.

Key length is the most important factor of encryption strength. For instance, 128-bit RC4 encryption is 3×10^{26} times stronger than 40-bit RC4 encryption. It is required that network nodes use at least 128-bit key for all symmetric encryptions.

7.3.3 Public Key Encryption

To avoid key distribution problems and make it easier to share confidential content with a number of people, asymmetric or public-key cryptography was designed. Public key encryption, first proposed in 1976 by Stanford professor Martine Hellman and his student Whitefield Diffie, uses a pair of related keys: public key for encryption and private key for decryption. The strength of public key encryption stems from the fact that when data is encrypted with one key, it can only be decrypted by its counter part. For instance, when data is encrypted using the public key, the corresponding private key is required in order to decrypt the data. Even if the public key is widely disseminated or intercepted by someone else, without the appropriate private key, this third party will be unable to decrypt the message. It is crucial that confidentiality can be supported without requiring that the sender and recipient exchange any secret key in addition to the message itself, as is the case in symmetric cryptography.

When a key pair is generated, the holder makes the public key available to everyone who wishes to send an encrypted message, and keeps the other part, the private key, as the secret. The following Figure 6 shows an encryption/decryption process using a key-pair:
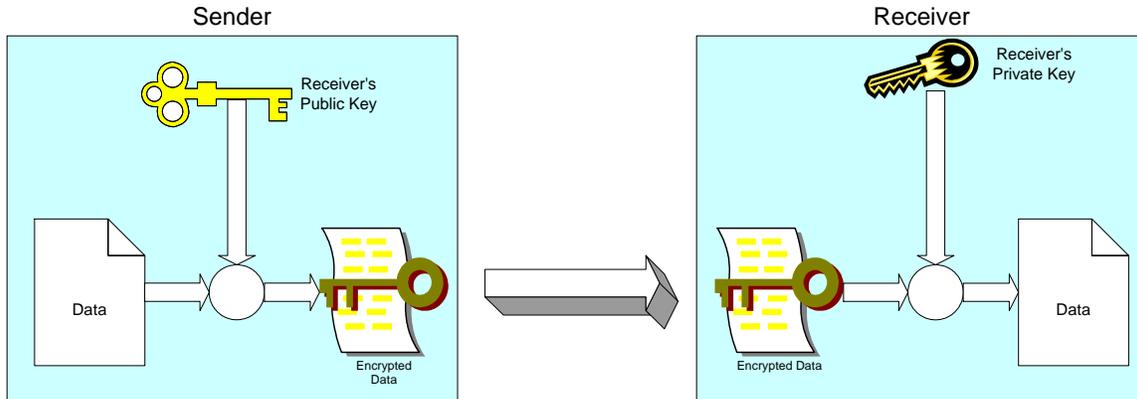

Figure 6: Public-Key Encryption and Decryption

The process is quite similar to the secret key encryption:

1. The sender encrypts data using the receiver's public key.

2. The sender sends the encrypted message to the receiver.

3. The receiver decrypts the data using the private key.

Public key algorithms solve two (2) major issues of the shared-secret algorithms:

1. Key Confidentiality: Since the private key never leaves the hands of the receiver, the confidentiality of the key is assured.

2. Key Distribution: The public key can be distributed using any communication channel without compromising the strength of encryption.

Compared with secret key encryption, however, public key encryption requires more computation and is therefore not always appropriate for large amounts of data. Secondly, locating a public key may become an issue if a network node is used by tens of thousands of users.

7.3.4 XML Encryption

XML Encryptions enables the transmission of data and tags using a standard encryption approach. XML Encryption depicted in Figure 7 serves the purpose of maintaining the confidentiality of information, both while in transit as well as when stored.

To provide a robust level of protection both a public and symmetric key should be used together. When both are used, the symmetric key encrypts the data and the public key encrypts the symmetric key. This requires that the public key cipher have higher strength than the symmetric key cipher - a broken symmetric key typically compromises only a single message, but a broken public-key pair compromises all messages to a given recipient.
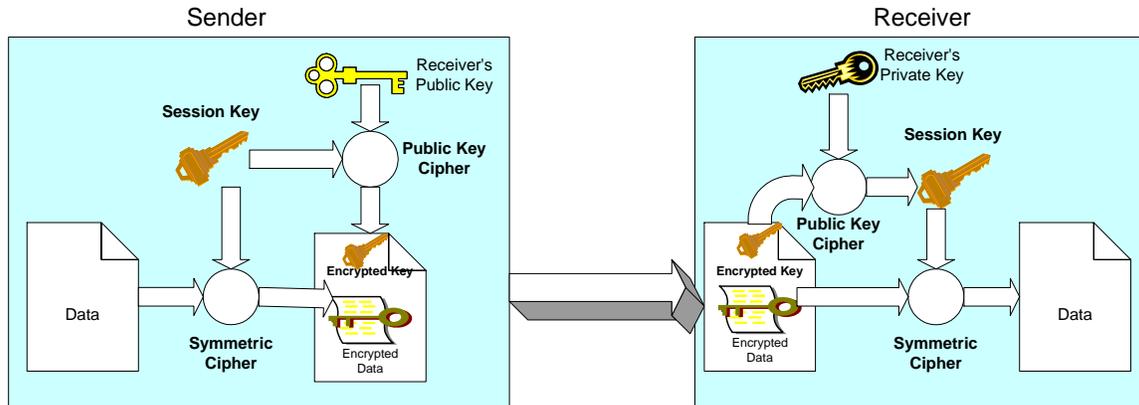
Figure 7: XML Encryption and Decryption

The process of XML Encryption is outlined below:

1. The sender generates a random session key.
2. Data objects are encrypted using a symmetric encryption algorithm with the session key.
3. The session key is then encrypted using the receiver's public key and inserted into the message as the key value.
4. The encrypted content, encrypted key and necessary algorithm information are packaged together.
5. The package is sent to the receiver.

The recipient may obtain the original content using the following steps:

1. Unpack the package to obtain the algorithm information, the encrypted symmetric key and the encrypted content.

2. Decrypt the symmetric key with their private key.

3. Decrypt the content with the symmetric key.

There are several advantages of this procedure:

- Data is encrypted using symmetric ciphers, which are more efficient and more suitable for large amounts of data.

- The session key is encrypted using public key algorithm with high strength.

- Session keys are generated randomly. It is almost impossible for adversaries to decrypt subsequent messages even if one session key is compromised.[17]

7.3.5 XML Encryption Example

The following XML segment contains an encrypted XML message and an encrypted session key:

```
<EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#">
    <EncryptionMethod Algorithm="http://www.w3.org/2001/04/
      xmlenc#tripledes-cbc" />
    <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
     <EncryptedKey xmlns="http://www.w3.org/2001/04/
        xmlenc#">
       <EncryptionMethod Algorithm="http://www.w3.org/
          2001/04/xmlenc#rsa-oaep-mgf1p" />
       <KeyInfo xmlns="http://www.w3.org/2000/09/
          xmldsig#">
         <KeyName>soapclient.com</KeyName>
       </KeyInfo>
       <CipherData>
         <CipherValue>DEXS4l+V5uPNacjfRZpC6rYJQdeRrsG5bWG
         PbL5LSs+bRaIS0ns7KJg42j8Jzk0W QqyuSreGAZTOv/
         bKfkL3kg==</CipherValue>
       </CipherData>
     </EncryptedKey>
    </KeyInfo>
    <CipherData>
     <CipherValue>4ubv13qV2Peycqra0MhSLZU3LnjCrDihtmGLaNhY+dV
     K+ QgwWjOP0o+y2mvDNfmWC/Fh/
     GD8bpBpfac1jEmGg7ngqvPGuj2Q+5A==</CipherValue>
    </CipherData>
</EncryptedData>
```

Encrypted Key

Encrypted Data

This example XML Encryption contains two (2) types of information:

1. Encrypted Data: Contained in the CipherData element.

2. Encrypted Key: Contained in the KeyInfo element. The key is encrypted using the receiver's public key.

7.4   Integrity and Non-repudiation

Any message on the Internet can be intercepted and modified. Message integrity ensures that the recipient of the message is notified, if the message has been tampered. Message integrity protection is a process of message authentication – proving a message is genuinely from its sender.

Repudiation is often involved with legal issues. An alleged sender of a document is always able to repudiate, or refute, a document that has been attributed to him or her. So non-repudiation is a way of proving that the sender has authorized, or approved, the document electronically and the document has not been forged.

The following sections describe how both message integrity and non-repudiation can be achieved using digital signatures. Digital signature relies on the uniqueness and privacy of private keys, i.e., there is only one owner of a private key that can be used to digitally sign a document. If the origin of the key is verifiable by a third party, then the signature is undeniable evidence of the signatory's approval to the document or message.

7.4.1 *Digital Signature*

There are three (3) key properties of a digital signature:

- Only someone possessing the private key can create the digital signature.
- Anyone who has the corresponding public key can verify the digital signature.
- Any modification of the signed data invalidates the digital signature.

Largely due to the performance characteristics of public-key algorithms, the entire message data is typically not itself signed directly with the private key. Instead, a digest of the document, also called a "hash", is signed.

Hash algorithms are one-way mathematical algorithms that take an arbitrary length document and produce a fixed length output string. A hash value is a unique and extremely compact numerical representation of a piece of data. MD5 produces 128 bits for instance. It is computationally improbable to find two distinct inputs that hash to the same value. Therefore, hash value is also considered the fingerprint of the document.

Because the hashing algorithm is very sensitive to any changes in the source document, it allows a recipient to verify that the document was not altered. Additionally, by signing the hash with the private key, the sender also allows the recipient to verify the signature. This is because there is only one such private key that could possibly match the public key.

Digital signatures authenticate the identity of a sender and protect the integrity of data.

Figure8 shows a general process of digital signature. It can be described as follows:

The sender:

1. Calculates a hash value of the data to be signed.
2. Signs the hash value using the sender's private key.
3. Embeds the signature into the document.
4. Sends the signed document to the receiver.

The receiver:

1. Decrypts the signature using the sender's public key. The result is a hash value calculated by the sender.
2. Calculates the hash value of the data.
3. Compares the two hash values. The signature is valid if the hash values match.
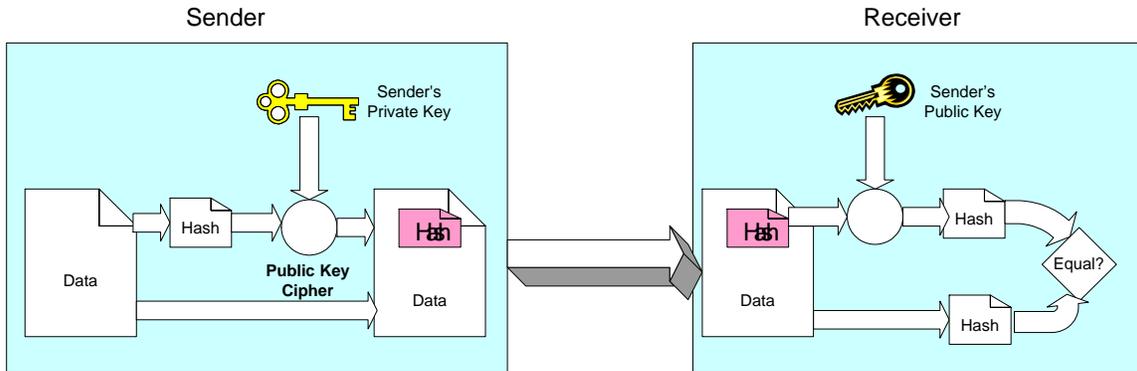
Figure 8: Process of Digital Signature

7.4.2 XML Signature

XML signatures are digital signatures designed for use in XML documents. XML signatures add authentication, data integrity, and support for non-repudiation to the data that they sign. However, unlike non-XML digital signature standards, XML signature has been designed to both account for and take advantage of the Internet and XML.

7.4.3 XML Signature Example

The following XML segment contains an XML signature:

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#" Id="mySignature">
  <SignedInfo>
     <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
         c14n-20010315" />
     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"
         />
   <Reference URI="#Body">
      <Transforms>
         <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
             c14n#WithComments" />
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <DigestValue>jYtbrxZ2xPNIO1PXBFpGmem1rwk=</DigestValue>
   </Reference>
  </SignedInfo>
  <SignatureValue>ZAZtXmYXhjB2iBn6amfl0KW5PBKjtSGuvzekGWaZU+wVFU6MeYv1
NcEW01q2403jrSoLZOEzr6X8Wi5O5SjDLWHnCoLDYPVHZcHFyZ/
bKFkg==</SignatureValue>
<KeyInfo>
    <KeyName>soapclient.com</KeyName>
  <KeyValue>
    <RSAKeyValue>
        <Modulus>l7j+tD+DNXgWiQTsK2GMv8RfAIFKRebZzeniPJc7Ra2q5o0Ld3E
        HAU98+X3iGardaaA7waMr/ZA Oo5CFCBhEh/j/AWxBdVIww==</
        Modulus>
      <Exponent>AQAB</Exponent>
    </RSAKeyValue>
  </KeyValue>
  </KeyInfo>
</Signature>
```

There are several key elements in the XML segment:

- **SignedInfo**: Contains what is signed (Reference) and how it is signed (SignatureMethod).

- **Reference**: Points to the portion of the XML document that is signed. An XML element with id of 'Body', perhaps a SOAP message body, is signed in this example.

- **DigestValue**: The hash value of the signed element.

- **KeyInfo**: Contains the signer's public key information.

Note that everyone could verify the signature because the signer has attached complete public key information, both the key name and the key value, in the document.

7.5 SOAP Message Integrity and Non-repudiation

SOAP message integrity can be protected using XML signature, which ensures that contents of a document were not tampered with during transition. Contrary to a popular belief that digital signatures offer more protection than encryption, signature and encryption are actually integral parts of one thing - information security. Encryption only hides contents of a document; the contents can still be altered during transition. On the other hand, a digitally signed document without encryption is similar to an open letter without sealing.

Another very important aspect of digital signature is Non-repudiation. Documents without digital signature are considered invalid by some data flows from a legal point of view. Digital signature is no longer an optional feature in such situations.

The WS-Security specification, proposed by IBM and Microsoft, defines a set of processes and rules that applications must follow in order to be compliant and interoperable. It is desirable that the SOAP stack provider supplies an implementation of WS-Security as part of the SOAP toolkit.

For messages with attachments, calculation of a digest should include all attached files. In other words, both the SOAP main message and attachments should be protected by signing a combined digest of all parts.

An alternative approach is to generate a signature for each individual part, body and attachments, and insert multiple signatures in the SOAP message header. This approach adds extra processing in the SOAP header, but allows more flexible signature verification.

Signatures, when present in a SOAP header, must have the mustUnderstand attribute set to "true". Validation of signatures by the receiver is mandatory.

8   Intrusion Detection

An Intrusion Detection System (IDS) is concerned with detecting break-ins after all other defensives, such as firewalls, virus scans, staff personnel screens, passwords, and so on have been compromised. Generally IDSs perform two types of functions:

- Detection of Abnormalities (DoA): which is concerned with system behavior that is a departure from what is detected normally.
- Detection of Misuse (DoM): which is concerned with scanning the systems for specific patterns of abnormal behavior.

DoA and DoM are push and pull approaches to screening for system problems. The DoM function, since it uses fixed patterns, has a low incident of false alarms. However, it can be compromised by new approaches to system break-ins. DoA functions can address this by detecting the break-in as a departure from normal operation. The down side of DoA is a higher level of false alarms. Successful IDS systems balance and merge DoA and DoM functions.[18]

IDS deployments can be classified[19] as follows:

- Network: where network traffic is monitored at the packet level to discover DoA and DoM behaviors when systems are being accessed.
- Host: where the host security system, user behavior, and applications are monitored. Mainly logging information is studied. The logging study function can be active (real time) or passive (batch).
- Blocking: where firewall rules are changed based on the behavior observed. This type of IDS usually works in conjunction with the host computer(s).
- Honeypot: where a system is setup to make it appear that it contains sensitive information. The system actually contains no sensitive information and it is used as a trap to catch unauthorized access.
- Enterprise: is a consolidation of individual IDS systems to create an overall enterprise view of intrusion detection.

IDS systems when combined with the system monitoring and management systems do offer the ability to dynamically stop intrusion when it occurs. Because the technology is newer, it should be applied with prudence or risk having production system access shut down due to false DoA readings.

9 Network Security Architecture

9.1 Security Strategies and Recommendations

9.1.1 Centralization

Given the number of network nodes in the Exchange Network and the diversity of IT environments at partner sites, it is unrealistic to require all network nodes to develop and deploy similarly strong security components. Therefore, it is necessary to develop components or Web services that can be shared or reused by all network nodes. "Centralization" is fundamental to reducing development and administrative costs, and complexity.

Although centralized services are resources to network nodes, their usage is not mandatory. Network nodes can still leverage existing security infrastructure and perform additional verification and validations if necessary.

9.1.2 Phased Deployment

It is clear that we cannot reach the goal of total security at once due to resource constraints and lack of commercial product support. We propose a staged implementation and deployment in the following order:

1. Network Authentication and Authorization Services (NAAS): These are centralized Web services that provide user authentication and access control. Initial implementation of the service may use a simple password authentication method and user-based authorization scheme. We recommend third-party SAML implementation as a plug-in component as the backend security processor for future deployment.

2. Central User Management Services: These services are required as the number of users of the Exchange Network grows. The user management services are considered network administrative components, and thus are not necessarily SOAP services. However, the services must allow states to manage their user identities and grant access to these users.

3. Central Key Management Services: We recommend the services be implemented using the W3C XKMS specification soon. The key management services are fundamental to XML signature and XML encryption.

4. Signature Support at CDX: The CDX node is the gateway of all EPA submissions. We expect wide support of XML signature in off-the-shelf commercial products in the next 6 to 12 months, which means requesters may easily sign documents. The CDX node must be able to verify signatures and assure document integrity.

5. XML Encryption/Decryption Support at CDX: Current SSL with authentication is sufficient for basic confidentiality requirements between parties. For data exchanges that require high-level privacy, XML Encryption is strongly recommended. Note that such exchanges are likely between requesters and the

CDX node. XML Encryption/Decryption support at the CDX node allows encrypted documents to flow into EPA's backend systems.

9.2 Key Network Security Services

Figure 9 shows a set of projected network services. The services are compatible with the Node Functional Specification, and can be developed independently with no impact to the existing network node interfaces.

All services in the cluster are XML Web services, which can be accessed easily by clients using the corresponding WSDL files.
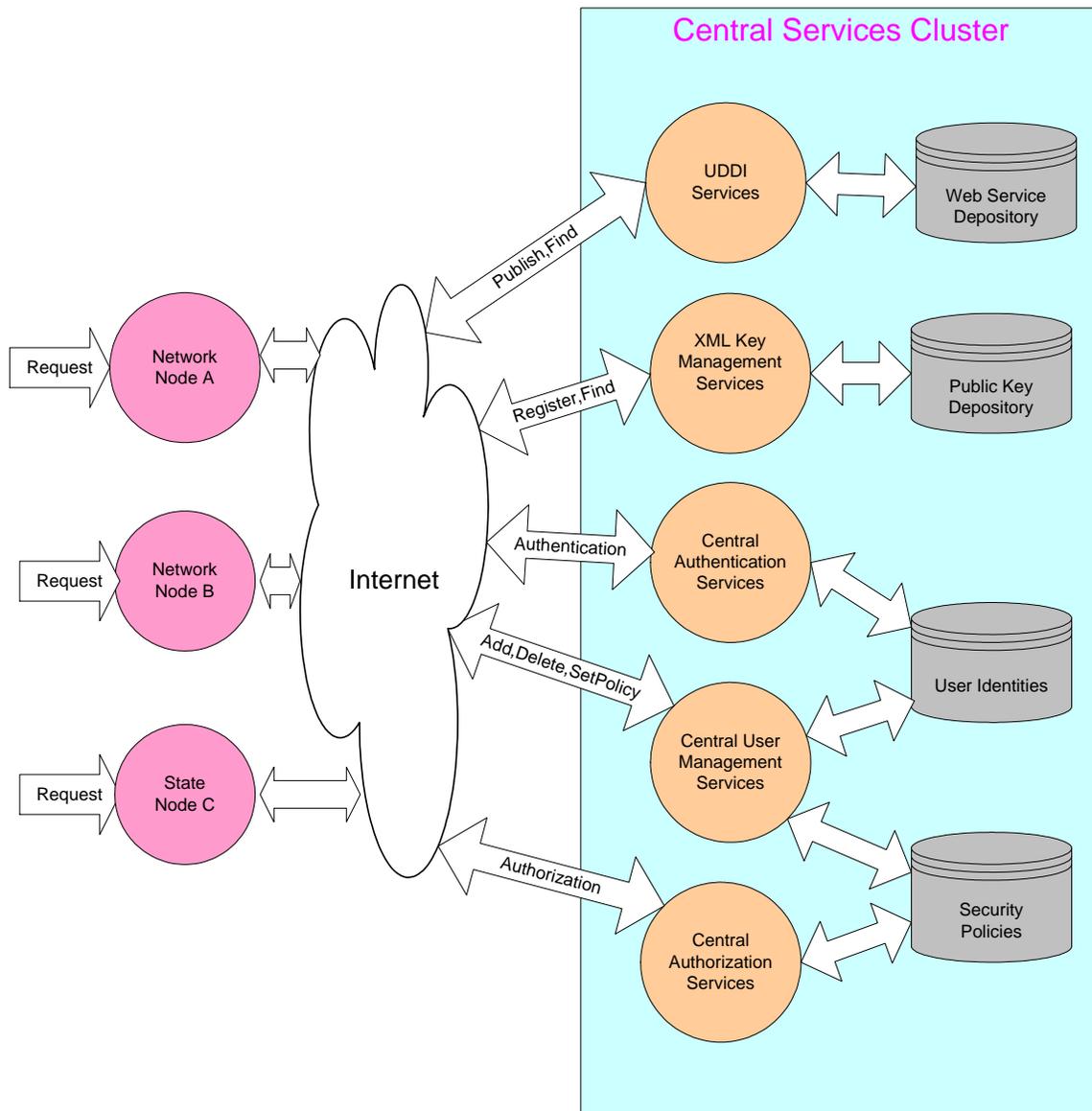
Figure 9: Architecture of Central Network Services

9.3 Central Authentication Services

The Central Authentication Services reduce the responsibility of implementing authentication features from state nodes, and makes them independent, loosely-coupled Web services accessible by any network node. This enhances the reusability, maintainability, extensibility and manageability of security resources within the Exchange Network.

The Central Authentication Services are available now to all network nodes as a set of Web services.

9.4 Central Authorization Services

The Central Authorization Services perform entitlement operations based on predefined security policies. Given the requested resource URL and the user identity, the service determines whether or not the operation is permitted.

9.5 Central User Management Services

This service is a set of Web services that allow network node administrators to add, remove and change user accounts. It also allows node administrators to grant access rights to network users through a simple Web service interface.

9.6 Central Key Management Services (CKMS)

Public key management is a critical component of XML signature and XML encryption. As the number of nodes grows, and the number of users increases, the Exchange Network needs a standard way to find, register and validate public keys. The CKMS is an elegant solution for the complex problem.

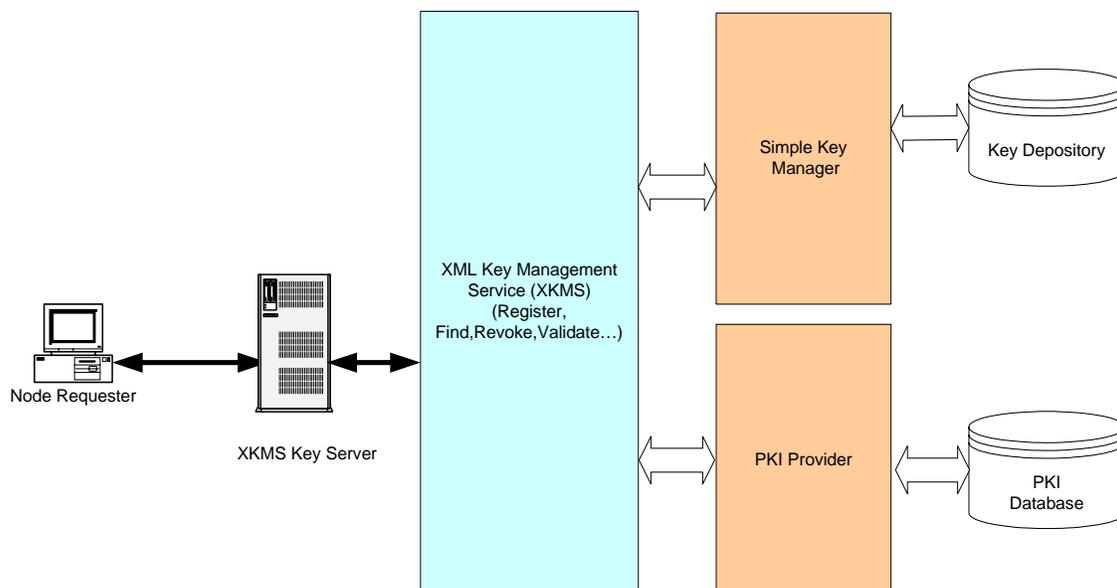The following Figure 10 shows a general structure of the CKMS:

Figure 10: CKMS General Structure

The service is based on XKMS and contains some key components:

- XKMS Services: This is the interface to all network nodes, which can be used to query, validate and find public keys.
- Simple Key Manager: This is a component that provides simple key services without certificates. The simple key management part is easy to implement and to deploy. It is expected to be part of the initial configuration of the XKMS services.
- PKI Provider: This is a provider of full implementation of PKI. Since PKI implementations are proprietary and expensive, the component is expected to be introduced later.

9.6.1 XKMS Functions

A XKMS service when interacting with the Simple Key Manager and PKI Provider supports three (3) primary operations:

- Registration: is used by the owner of the keys to manage them. The base register operations include register, reissue, revoke, and recover.

- Locate: is used to locate public keys from a trusted server. The locate operation returns the public key to the caller.

- Validate: is used to validate the binding of a key return by the Locate. This service is sometimes integrated with a Domain Name Service (DNS). When using DNS integration the public key is bound to an address (like an E-mail address). The Validate is called by the user application using the bound address. The Validate using the DNS finds the Locate service, retrieves the public key, and validates it.

The XKMS service does not have to use a PKI Provider since the key service is masked by the above operations. It is possible to have the XKMS service application generate and manage the keys directly. This feature should create deployment options and provide for greater flexibility when using XKMS.[13,20]

9.6.2 The Benefits of XKMS

Some of the benefits XKMS provides[13]:

- **Simple Interface**: In the spirit of SOAP simplicity, the complexities of key management is masked behind three (3) services: Registration, Locate, and Validate. Applications do not have to deal with the complexities of PKI.

- **Technology Neutrally and Flexibility**: XKMS does not define the type of security used, but provides the mechanisms to present the key management data in a non-proprietary format to the application. Security technologies can be changed without impacting applications.

- **Small Footprint**: Client applications are simpler due to the simplicity of the interface to the XKMS. This results in smaller application runtimes when they use XKMS services.

- **Implementation Flexibility**: With XKMS using interfaces to the security services, different packages could be plugged in to provide the Registration, Locate, and Validate services.

10 Conclusions

It is clear that the current network environment provides adequate support for data exchanges that do not require non-repudiation or document-level confidentiality. The authentication requirement defined in the Node Functional Specification Version 1.0 and the basic channel encryption provided by SSL can effectively prevent unauthorized access on the network.

However, it is expected that some of the data flows, which are highly sensitive and require digital signature, will require much stronger security features. These features can be provided as either separate services (such as XKMS) or as add-ons to the existing nodes (such as XML Signature).

As WS-Security specification is being turned over to OASIS, we expect it to become a widely-adopted Internet standard supporting implementations on different platforms. Microsoft has recently released a Web Service Enhancement (WSE) 1.0, which includes support to the current WS-Security specification; and VeriSign plans to release an open source implementation of WS-Security soon. IBM includes a Java implementation in its Web Service Toolkit (WSTK).

We strongly recommend that the CDX node add WS-Security support as soon as possible. The CDX node is a critical point for all data flows; both XML encryption and XML signature are considered essential, not optional.

Web service security has been, and will continue to be, the primary concern of SOAP-based applications. Current technologies and available products are sufficient for building significantly secure network nodes. Armed with NAAS, WS-Security and other centralized security services, the Exchange Network can be used to exchange all EPA documents including those with a high sensitivity level.

11 References

[1]W3C Note 8, *SOAP: Simple Object Access Protocol 1.1*, May 2000. Copyright © 2000 DevelopMentor, International Business Machines Corporation, Lotus Development Corporation, Microsoft, UserLand Software: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[2] *Network Node Functional Specification*, Version 1.0, March 14, 2003.

[3] M. Hondo, N. Nagaratnam, and A. Nadalin, *Securing Web Services*, IBM Systems Journal, Vol. 41,No 2, 2002: http://www.research.ibm.com/journal/sj/412/hondo.pdf

[4] D.F.C. Brewer, M.J. Nash, *The Chinese wall security policy*. In Proceedings of IEEE Symposium on Security and Privacy, 1989, pp. 206-214.

[5] Vijayalakshmi Atluri, Soon Ae Chun, and Pietro Mazzoleni, *A Chinese Wall Security Model for Decentralized Workflow Systems*, November 5-8, 2001, ACM CCS'01: http://cimic.rutgers.edu/~atluri/ccs01.pdf

[6] Carl Ellison, Bruce Schneier, *Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure*, Counter Pane Computer Security Journal, Volume XVI Number 1, 2000: http://www.counterpane.com/pki-risks.pdf

[7] IBM, Microsoft, *Security in a Web Services World: A Proposed Architecture and Roadmap*, April 7, 2002: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwssecur/html/securitywhitepaper.asp

[8] W3C Recommendation 10, *XML Encryption Syntax and Processing,* December 2002, Copyright © 2002 W3C® (MIT, INRIA, Keio), All Rights Reserved.: http://www.w3.org/TR/xmlenc-core/

[9] Matt Powell, *WS-Security Authentication and Digital Signatures with Web Services Enhancements*, December 2002: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwssecur/html/wssecauthwse.asp

[10] OASIS Standard, *Web Services Security: SOAP Message Security*, Working Draft 11, March 3, 2003, Copyright © OASIS 2002. All Rights Reserved.

[11] OASIS Standard, *eXtensible Access Control Markup Language (XACML) 1.0*, February 18, 2003, Copyright © OASIS Open 2003. All Rights Reserved.: http://www.oasis-open.org/committees/xacml/repository/oasis-xacml-1.0.pdf

[12] OASIS Standard, *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*, November 5, 2002, Copyright © OASIS 2001, 2002. All Rights Reserved.

[13] W3C Note 30, *XML Key Management Specification (XKMS)*, March 2001, Copyright © 2001 VeriSign Inc, Microsoft Corporation, webMethods Inc.: http://www.w3.org/TR/xkms/

[14] ASN.1 Information Site: http://asn1.elibel.tm.fr/en/index.htm

[15] Don Box, et al. *Web Service Policy Framework (WS-Policy)*, December 18, 2002.

[16] L. Wang, D. Wijesekera, and S. Jajodia, *Toward Secure XML Federations*, to appear in 16th IFIP WG11.3 Working Conference on Database and Application Security (IFIP-WG11.3 2002).

[17] W3C Proposed Recommendation 20, *XML Signature Syntax and Processing*, August 2001, Copyright © 2001 The Internet Society & W3C® (MIT, INRIA, Keio), All Rights Reserved.: http://www.w3.org/TR/2001/PR-xmldsig-core-20010820

[18] Anup K. Ghosh, Aaron Schwartzbard, and Michael Schatz, *Learning Program Behavior Profiles for Intrusion Detection*, April 9-12, 1999, Proceedings of the 1st Workshop on Intrusion Detection and Network Monitoring: http://www.usenix.org/publications/library/proceedings/detection99/full_papers/ghosh/ghosh_html/

[19] Nikolai Bezroukov, *Network-level Intrusion Detection*, 1999: http://www.softpanorama.org/Security/network_ids.shtml

[20] W3C Working Draft 18, *XML Key Management Specification (XKMS 2.0) Part I: Schema*, February 2003, Copyright © 2002 W3C® (MIT, INRIA, Keio), All Rights Reserved.: http://www.w3.org/2001/XKMS/Drafts/XKMS20030212/xkms-part-1.html