SECURITY CONSIDERATIONS IN A GLOBAL MESSAGE SERVICE HANDLER DESIGN (RESEARCH IN PROGRESS)

Johannes J van Eeden[a] and Maree Pather[b]

[a] Port Elizabeth Technikon – Author

[b] Port Elizabeth Technikon - Co-author

[a] s20036227@student.petech.ac.za

[b] maree@petech.ac.za

ABSTRACT: Web services are generally accepted as the most interoperable application interface today on the Web. In the context of a global electronic marketplace this is an essential factor. In keeping with Services-Oriented Architecture trends, a Web service-based Message Service Handler can provide a global service to all participants in the global marketplace. The main objective of this research is to design a Web service to provide Message Handler Services, using ebXML as the point-of-departure. The focus of this paper is to arrive at a set of pre-specified security standards to promote the goal of interoperability, explaining, with justification, which security mechanisms should be used within the proposed Web service model. The Web service will send messages using the SOAP with Attachments architecture. The use of XML signatures and XML encryption within this SOAP envelope is advised to ensure integrity, authentication and confidentiality. When the actual SOAP envelope is transmitted over the Internet, it will be wrapped within an IPSec packet to ensure further security.

KEY WORDS: Message Service Handler, Web service, ebXML, IPSec, TLS/SSL, XKMS, XACML, SOAP, XML Encryption, XML Signatures, WS Security

SECURITY CONSIDERATIONS IN A GLOBAL MESSAGE SERVICE HANDLER DESIGN (RESEARCH IN PROGRESS)

1 INTRODUCTION

In a world where more and more business transactions occur electronically, using the Internet as a transport medium, interoperability and security have become very important. Interoperability provides for the seamless integration and interoperation between internal and external enterprise applications which might consist of autonomous, heterogeneous, and distributed components that form part of loosely coupled and/or tightly coupled systems (Bouguettaya et al, 1998). To promote interoperability, the use of the same protocols and standards are important to allow different systems to communicate with each other. Web services have become very popular and can been seen as a solution to address interoperability issues.

Fundamentally, a Web service can be seen as a service that is accessible over the Internet. However, the term is more far-reaching and refers to the architecture, standards, technology and business models that make a Web service possible. IBM (2001) defines a Web service as a new breed of Web applications. It is stated that they are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. Gardner (2001) states that Web services are interoperable building blocks which are used for constructing applications. The Web services architecture enables applications to connect to other applications.

Gardner (2001) states that Web services architectures need to address discovery, authentication, authorisation, and business models for accessing intellectual property. Another issue is the interaction with services once they are accessed. One needs to know how to present the resource such that these issues are addressed. Gardner (2001) states that the infrastructure for presenting the resource is well established; it involves sending a MIME type with a document and then using an application that knows how to deal with that document type. A problem which was brought up with services is that there are potentially many more types of services than types of document and that the interactions with Web services are more complex. The Web services architecture therefore requires clients to know in advance the type of service that they will be using.

To ensure interoperability, certain standards have been developed. These standards will be described in the next section. In section 3, the purpose of a Message Service Handler (MSH) is discussed, followed by a proposed model explaining the need for a MSH in a global electronic marketplace.

2 INTEROPERABILITY AND SECURITY STANDARDS

In this section the focus is on standards that are available to ensure interoperability and the required security measurements.

2.1 Web Service Description Language (WSDL)

Communication protocols and message formats are becoming more standardized in the Web community. The need arises for the importunateness that these communications must be described in some structured way. WSDL (2001) addresses this need by defining an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. Within these XML grammar definitions, one will find documentation for distributed systems which serve as a recipe for automating the details involved in application communications.

2.2 Universal Description, Discovery and Integration (UDDI)

A Web service can only be meaningful if potential users can find appropriate information about how to execute them. According to the UDDI Specification (UDDI, 2002) the focus of UDDI is the definition of a set of services supporting the description and discovery of businesses, organizations, and other Web services providers, the Web services they make available, and the technical interfaces which may be used to access those services. UUDI is based on a common set of industry standards such as HTTP, XML, XML Schema, and SOAP. UDDI provides an interoperable, foundational infrastructure for a Web services-based software environment for both publicly available services and services only exposed internally within an organization.

2.3 Simple Object Access Protocol (SOAP)

SOAP is described by Medjahed et al (2003) as a lightweight messaging framework for exchanging XML-formatted data among Web services, which functions using a variety of transport protocols (e.g. HTTP, SMTP, FTP). Web services use SOAP as a messaging protocol. This adoption of XML-based messaging facilitates interaction between heterogeneous systems.

Drummond (2001) explains that the SOAP framework begins with a normal MIME envelope within which a XML namespace-qualified SOAP envelope is encapsulated. ebXML extends this MIME structure by using the multipart/related framework used by SOAP with Attachments to include additional payload data.

A major design goal for SOAP is simplicity and extensibility, according to the SOAP Specification (SOAP, 2000). SOAP messages are fundamentally one-way transmissions from a sender to a receiver, but SOAP messages are often combined to be used as request/response messages. SOAP messages can be set to many different exchange patterns e.g. one way, request/response, multicast, etc.

In terms of security, SOAP does implement some methods for integrity and privacy protection; primarily, signatures are used within messages.

2.4 SAML

While current technologies enable an e-business to authenticate users and manage user access privileges, it takes considerable effort and cost to extend these capabilities across an enterprise or share them among trading partners. According RSA (2003), the Security Assertions Markup Language (SAML) addresses this challenge. SAML is an XML-based framework that enables Web services to readily exchange information relating to authentication and authorization. This information takes the form of trusted statements, called security assertions, about end users, Web services, or any other entity that can be assigned a digital identity.

There are three major types of SAML assertions.

- Authentication assertions, which are issued by an authentication service, declare that the identity of a user or a Web service has been authenticated to access protected resources, for example, an intranet or extranet.
- Attribute assertions, generated by an attribute service, verify that a user or Web service possesses certain static attributes (e.g., job role or company affiliation) or dynamic attributes (such as a consumer's bank account balance or a reseller's quarterly sales volume to date). Attribute information is vital to the process of assigning Web access privileges.
- An authorization service brings together authentication assertions, attribute assertions and authorization policies and generates authorization assertions that define which resources a user/service is entitled to access. Because SAML shields applications from the complexity of the underlying authentication and authorization systems, it has the flexibility to address a range of interoperability challenges, both today and in future environments. SAML is

already being used for authorization in higher education institutions and on the public Internet.

2.5 XACML

Griffin (2004) states that the Extensible Access Control Markup Language, or XACML promises to standardize policy management and access decisions. XACML can be used to define a general policy language to protect resources as well as act as an access decision language. Therefore, if a user wants to access a resource, an authorization process is performed. This makes it a key component in the development of authorization infrastructures and a foundational step in the creation of federated authentication environment.

Olavsrud (2003) states that according to Sun, XACML has a number of advantages over other access control policy languages, including:

- One standard access control policy language can replace dozens of application-specific languages
- Administrators save time and money because they do not need to rewrite their policies in many different languages
- Developers save time and money because they do not have to invent new policy languages and write code to support them; they can reuse existing code
- Good tools for writing and managing XACML policies will be developed, since they can be used with many applications
- XACML is flexible enough to accommodate most access control policy needs and extensible so that new requirements can be supported
- One XACML policy can cover many resources; this helps avoid inconsistent policies on different resources
- XACML allows one policy to refer to another; this is important for large organizations, for instance, a site-specific policy may refer to a company-wide policy and a country-specific policy.

A downfall mentioned by Griffin (2004), is the fact that XACML can increase system overhead; in an example given - a Web page which aggregates many resources – an XACML request is required for each resource.

2.6 XKMS

It is stated by Salz (2003) that public-key infrastructures (PKI's) are well suited for securing Web services, but PKI deployment is too cumbersome and costly for the technology to achieve widespread use. The XML Key Management Specification borrows the best of PKI without reducing scalability or security. According to XML Trust Centre (2004), the purpose of XKMS is to define a Web services interface for a public key infrastructure. This makes it easy for applications to interface with key-related services, like registration and revocation, and location and validation. Developers only need to implement XKMS clients, because XKMS server components are mostly implemented by PKI providers. XKMS is a solution for secure Web services, enabling Web services to register and manage cryptographic keys used for digital signatures and encryption.

2.7 WS-Security

The WS-Security Specification (2004) (Web Services Security), proposes a standard set of SOAP extensions that can be used when building secure Web services to implement integrity and confidentiality. This provides quality of protection through message integrity, message confidentiality, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies. WS-Security is flexible and is designed to be used as the basis for the construction of a wide variety of security models. The specification provides three main mechanisms, namely security token propagation, message integrity, and message confidentiality. By themselves, these mechanisms do not provide a complete

security solution. Instead, WS-Security is a building block that can be used in conjunction with other Web service extensions and higher-level application-specific protocols to accommodate a wide variety of security models and encryption technologies.

2.8 XML-Signatures

In an article published by Simon, Madsen, and Adams (2001), it is said that security technologies implemented by common developments are not enough for securing business transactions over the Web. Browser security is not enhanced or flexible enough to protect highly sensitive B2B transactions. Transport security protocols, such as Transport Layer Security/Secure Sockets Layer (TLS/SSL), are sufficient to protect a transaction while en-route, but the transaction components are not protected when it is stored on a server (which might be public). The need exists for message authentication, integrity and non-repudiation. The globally recognized method for satisfying these requirements for secure business transactions is to use digital certificates to enable the encryption and digital signing of the exchanged data. XML signatures are digital signatures designed for use in XML transactions. A useful feature of XML-signatures is that it enables one to sign selective parts of a message. Different sections of an XML document can be signed by different signatures (Simon et al., 2001).

2.9 XML-Encryption

Using XML Encryption, certain parts of the message can be encrypted. Siddiqui (2002) states that both encrypted and non-encrypted data can be exchanged within the same document. Another benefit of XML Encryption is that it can handle both XML and non-XML (e.g. binary) data. The encryption is done with symmetric encryption after a secret key is generated using asymmetric encryption. The encryption of certain parts of a message is important when confidential information must not be revealed to certain intermediaries when the message is processed.

XML Signatures and XML Encryption work on the same basic principle.

2.10 IPSec

IPsec, short for IP Security, is a set of open standard protocols developed by the Internet Engineering Task Force IETF to support secure exchange of packets at the IP layer (network layer)(Cisco Systems, 2002). This provides security for transmission of sensitive information over unprotected networks such as the Internet. IPsec has been deployed widely to implement Virtual Private Networks (VPNs).

Tunnelling can be achieved using the Authentication Header (AH) protocol or Encapsulating Security Payload (ESP). Encryption can be enforced as high as 256-bit AES (Nortel Networks, 2003). Two encryption modes exist (Cisco Systems, 2002). In tunnel mode, both the headers and payload are encrypted. In transport mode, IP headers remain unencrypted, but payloads are encrypted. (In tunnel mode, the inner IP header is also encrypted.) However, with transport mode the risk of traffic analysis is possible.

2.11 TLS/SSL

TLS/SSL is a popular protocol for securing HTTP traffic. The TLS/SSL protocol suite encrypts communications between Web servers and Web browsers for tunnelling over the Internet at the application layer. TLS/SSL make use of standards-based encryption and authentication, and provide secure access to data and applications over the Web (Nortel Networks, 2003). No specific software is required to make use of TLS/SSL, because it is generally integrated into an application, e.g. a Web browser.

It is stated by Kurlekar (2003) that TLS/SSL uses public-and-private key encryption, which also includes the use of digital certificates. Two-way authentication is not available. This result in anyone who has a correct username and password being able to access a TLS/SSL virtual private network.

Problems with TLS/SSL are that it affects network throughput, because cryptographic processing is very much CPU-intensive, and TLS/SSL is not protected against traffic analysis. (Canavan, 2001).

3 MESSAGE SERVICE HANDLER BASED ON THE EBXML MODEL

Kiely (2001) writes that the main aim of Electronic business eXtensible Markup Language (ebXML), which is supported by the United Nations as a standard for E-business, is to create a single online marketplace where companies of any size or nationality can collaborate and conduct business around the globe. ebXML can be thought of as the successor to electronic data interchange (EDI). ebXML specifies common business processes and an architecture for carrying out those EDI processes over the Internet. ebXML provides interoperability within and between ebXML-compliant trading-partner applications and maximizes efficiency.

The Message Service Handler (MSH) in ebXML handles all the incoming and outgoing messages. All communication within ebXML must take place via the MSH. The MSH interfaces with an application that processes/generate messages. The MSH will perform authentication, authorisation, encryption and packaging actions on messages and sends them over specified protocols to a receiving MSH. The MSH also provides a means for error-handling.

In the current ebXML specification, it is not mandated how the MSH must be implemented. It is assumed that not all the functionality has to be implemented. All MSH implementations must comply with functions supported in a corresponding MSH, to enable them to communicate with each other. If a function is not implemented, the MSH must provide an error notification stating that the functionality is requested but not supported.

In the ebXML Message Specification (ebMS, 2002), it is dictated that the ebXML Messaging Service be conceptually broken down into the following three parts:

1. an abstract Service Interface
2. functions provided by the MSH
3. mapping to underlying transport services

Figure 1 shows a logical arrangement of the functional modules existing within one possible implementation of the ebXML Message Services architecture. These modules are arranged in a manner to indicate their inter-relationships and dependencies.

The Header Processing module handles the creation of the ebXML Header elements for the ebXML Message, using input from the ebXML application, passed through the Message Service Interface, information from the agreed Collaboration Protocol Agreement (CPA) governing the message and generated information such as digital signature, timestamps and unique identifiers.

The Header Parsing process extracts or transforms information from a received ebXML message's Header element into one or other form suitable for processing by the MSH implementation.

Security Services: the message service handler can include digital signature creation and verification, encryption, authentication and authorization. Security services can be

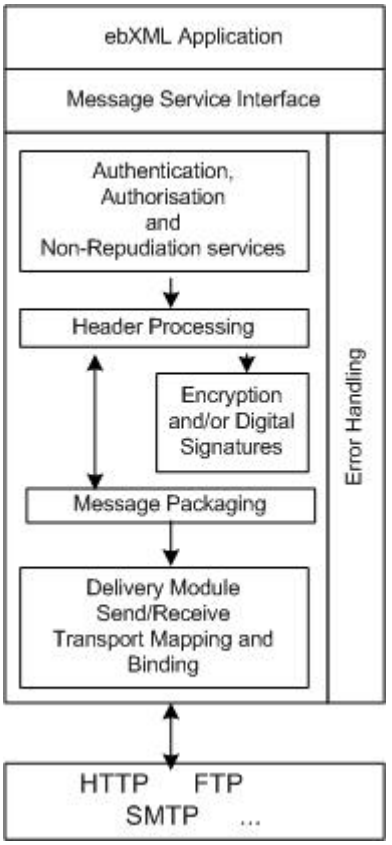

Figure 1: Relationship between ebXML MSH Components (ebMS, 2002)

implemented by other components of the MSH including the Header Processing and Header Parsing components.

The Reliable Messaging Service is responsible for the delivery and acknowledgment of ebXML Messages. The service includes handling for persistence, retry, error notification and acknowledgment of messages requiring reliable delivery.

The Message Packaging module in the message service handler is where the final enveloping of an ebXML Message (ebXML header elements and payload) into its SOAP Messages with Attachments container occurs.

The Error Handling component handles the reporting of errors encountered during MSH or Application processing of a message.

A Message Service Interface is an abstract service interface applications used to interact with the MSH to send and receive messages and which the MSH uses to interface with applications handling received messages.

Stamps (2003) lists the following as design goals for the MSH. It should be:

- Based on standards: XML, XSLT, HTTP, SMTP, SOAP
- Flexible and adaptable processing
- An open framework for customer-specific processing
- Use native XML as much as possible
- Allow integration of existing middleware
- Support legacy integration
- Provide encryption and signature support (XML signature, XML encryption)
- Provide easy security integration
- Support access to repositories (UDDI, ebXML repository)

4 A PROPOSED MODEL

The main goal of this research is to design a Web service that will facilitate messaging in a global electronic market, ensuring interoperability. As mentioned in section 3, the implementation of the MSH is not mandated. This allows developers to create implementations according to their own will, which will lead to interoperability issues, because each MSH will be based upon different standards. By creating a Web service which facilitates the functionality of the MSH, interoperability issues will be less, developers do not need to create their own message handlers (Web services act as building blocks for creating applications) and the service will be based upon the most appropriate standards. The standards discussed in this paper will primarily be considered within this proposed Web service. The use of specific standards and protocols will be suggested; implementations of gateways, which are used to translate between different protocols, can be very costly and could lead to further interoperability problems.

The Web service will essentially perform message handling during electronic business transactions. It will be responsible for the generation of a well-structured message envelope, including the necessary security measures to guarantee the safety and validity of a specific message.

The message handling Web service will require having certain specific features – some of which will mentioned briefly, as this research is in a very early stage. For instance, the Web service must be downloadable onto a company's network to be integrated into their trading system. An adversary might want to create a denial of service attack on a specific company's system; therefore methods must be available to counter such attacks. Digital signatures can be used to validate a

message. If messages are received from the same IP address and the same signature, those messages must be able to be filtered by a firewall or a similar countermeasure procedure.

If a company wants to send a message to a trading partner, an XML document will be created by an upper-layer application (which might be another Web service) that does not form part of the proposed Web service. If it is considered that security must be enforced within the document, it is solely up to this upper-layer application which creates the document to implement this level of security. For example, the upper-layer application might use WS-Security, SAML, XACML, XKMS, XML signatures and XML encryption within the XML document, but the proposed Web service will only accept the XML documents and attach it to the message. The Web service must be able to validate these documents according to a public XML schema document that represents a specific industry component. This XML document will be placed within a secure SOAP envelope for transmission.

Once the Web service is developed, the functionality and interaction with a service need to be described so that users know how to interact with these services. WSDL and UDDI are standards developed to describe and present the services.

As mentioned in section 3, the use of XML is recommended within a MSH, because of its flexible structure. The SOAP framework extends this flexibility, by allowing developers to create their own methods. Within the proposed Web service, these methods will be restricted to providing basic connection, choreography details, specified security measurements and the provision of MIME attachments.

Figure 2 shows a proposed SOAP message architecture. Italics indicate attributes and normal font indicates elements. In the SOAP Header, the MessageHeader element includes the version attribute, which indicates an *id* for a schema document that was used to create the attached message. These documents will be stored on a registry/repository. All the elements and their contents must be understood by the receiving process otherwise an error message will be created. ConversationId is a string identifying related messages that are part of a conversation. The service element specifies the service that must act on the message and the action element specifies the process of the service that must process the message. The service and action element values must conform to those specified in the business processes which are stored on the (ebXML) repository. MessageData uniquely identifies the (ebXML) messages. DuplicateElimination will prevent messages from being processed more than once. An MSH has to keep track of messages received in order to eliminate already received messages. The description element is a textual description of the message's intent. The signature element contains the XML digital signature of the sender. The CPAId element, which is a reference to the Collaboration Protocol Agreement (CPA), will be omitted from the message architecture, because only certain protocols are allowed to be used, therefore it is not necessary to have a CPA between trading partners. This protocol agreement, according to ebXML standards, is created by combining two companies' CPP's (Collaboration Protocol Profiles). These CPP's contain information about standards used by those specific companies. The CPA is a mutual agreement on the same standards to be

Figure 2: Proposed SOAP message architecture

used. If there is no agreement on which standards are going to be used, no communication will be able to take place. Using the proposed Web service, it will not be necessary to create CPA's.

The SOAP body includes one element, the manifest. The manifest element can contain multiple reference elements that link to the message payloads with textual descriptions of each payload. These payloads are possible by using the SOAP with Attachments standard.

To secure the SOAP message, the WS-Security recommends XML Signatures and XML Encryption. The message service handler will implement XML Signatures and encryption for the entire message. XKMS will be used for the management of the relevant keys.

The SOAP message generated by the Web service will be transported using TCP, with HTTP as the main application layer protocols.

HTTP 1.1 is a synchronous protocol that attempts immediate delivery. If delivery is not possible, an error will occur. An HTTP-server might be unavailable, because it is down or has too many queries to process. The sender will have to check regularly for availability, and this can lead to high overhead.

To avoid problems like this, it would be convenient to implement a message queue where messages will be stored until they can be processed. This can be accomplished by implementing HTTP with message-queuing-and-forwarding where HTTP will act as an asynchronous protocol.

The proposed Web service will cater for error-checking, error-handling and auditing at application level.

In ebXML, the digital signatures are implemented to protect the integrity and origin of messages (ebMS, 2002), but there are still vulnerabilities present. The impact of the vulnerability depends on the deployment environment and the transport mechanisms used to exchange these messages.

MIME is used as the framework for the message package, containing the SOAP envelope and any payload containers. Various SOAP envelope elements make reference to the payloads identified via the MIME mechanisms. The MIME Content-ID: header is used to specify a unique, identifying label for each payload. The label is used in the SOAP Envelope to identify the payload whenever it is needed. The MIME headers are not protected, even when an XML-based digital signature is applied. An ebXML message may be at risk depending on how the information in the MIME headers is processed as compared to the information in the SOAP Envelope. The Content-ID: MIME header is critical. An adversary could easily mount a denial-of-service attack by mixing and matching payloads with the Content-ID: headers. As with most denial-of-service attacks there are no specific protections offered for this vulnerability. An adversary could change the MIME headers while a message is en route from its origin to its destination and this would not be detected when the security services are validated. This threat is less significant in a peer-to-peer transport environment as compared to a multi-hop transport environment.

However, the solution seems to be relatively simple. All messaging using TCP is generally packaged in an Internet Protocol (IP) packet. By implementing Secure IP (IPSec, the VPN protocol of choice) at the network layer, secure "tunneling" between trading partners can be ensured. HTTP messages - which may include SOAP messages, with or without MIME attachments (SOAP payloads) will then travel in this secure tunnel. The MIME headers would then be protected within an IPSec packet.

In IPSec tunnel mode the whole message is encrypted, whereas using the transport mode, only the payload is encrypted. The secure tunnel mode should, ideally, be used to send messages between trading partner, because then adversaries would not be able to obtain information from the MIME and SOAP Headers. This type of encryption results in node-to-node encryption. However, each receiving device would have to be an IPsec-compliant device to decrypt each packet. If the

package is not at its destination it must be re-encrypted and forwarded to the next receiving device. Each receiving device has to have the sender's public key for its digital certificate. Thus, transport mode, with the traffic analysis vulnerability, is the more practical choice (providing end-to-end encryption). IPSec provides privacy, authentication and data integrity of IP packets (Phaltankar, 2000).

IPSec is considered by Budd and Gray (2002) to be more secure than TLS/SSL. They also state that IPSec can make use of maximum 168-bit Triple DES, whereas TLS/SSL uses maximum 128-bit RC4 encryption. It is said furthermore that TLS/SSL provides an inferior implementation of authentication than IPSec, because it makes use of client software. A feature of IPSec is that new algorithms, e.g. 256-bit AES encryption, can be added as they are developed.

5 CONCLUSION

In this paper the focus is on designing a secure Web service that will facilitate messaging within a global electronic business market.

A message structure is envisaged that will contain the business documents as payloads (refer to section 4) which is possible through the implementation of the SOAP with Attachments standard. Actual SOAP messages will carry business document payloads and be sent through an IPSec secure tunnel. XML Signatures and Encryption at the application level will be recommended to ensure security within the message payload. The IP-packets containing the SOAP message will be secured to ensure authentication, confidentiality, data integrity, non-repudiation and anti-replay protection.

Much research still needs to be conducted, but a design is being planned that will create an enhanced Message Handler Web service, that implements the most appropriate industry standards. A generic template from which configuration items may be selected for processing by both ends is being considered in order to make the Message Handler service universally applicable.

6 REFERENCES

Bouguettaya, A., Benatallah, B., & Elmagarmid, A. (1998). *Interconnecting heterogeneous information systems.* Boston, Mass., USA: Kluwer.

Budd, F., & Gray, D. (2002). *Security - A Focus on IPSec.* (Available at http://www.wiltelcommunications.com/vpn)

Canavan, J. (2001). *Fundamentals of Network Security.* Artech House.

Cisco Systems. (2002). *IPSec network security.* (Available at http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113t/%113t_3/ipsec.htm#xtocid0)

Drummond, R. (2001). Put SOAP and ebXML to Work. *E-Business Advisor Magazine*. (Available at http://advisor.com)

ebMS. (2002, Feb). *Message Service Specification Version 2.0 rev C.* (Available at http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf)

Gardner, T. (2001). Introduction to Web Services. *Ariadne, Issue 29*. (Available at

http://www.ariadne.ac.uk/issue29/gardner/)

Griffin, P. (2004). *Web services - introduction to XACML*. (Available at http://dev2dev.bea.com/technologies/webservices/articles/xacml.jsp)

IBM. (2001). *Web services – the Web's next revolution.* (Available at http://www-105.ibm.com/developerworks/ education.nsf/webservices-onlinec%ourse-bytitle/BA84142372686CFB862569A400601C18?OpenDocument)

Kiely, D (2001). E-business Language of Choice? *Information Week, 836*, 79.

Kurlekar, A. (2003). *Enterprise Basics: IPSec vs. SSL VPN*. (Available at http://www.esj.com/news/article.asp?editorialsId=539)

Medjahed, B., Benatallah, B., Bouguettaya, A., Ngu, A., & Elmagarmid, A. (2003). Business-to-business interactions: issues and enabling technologies. *The VLDB Journal*, *12*, 59+.

Nortel Networks. (2003). *Ipsec and ssl: Complimentary solutions.* (Available at http://a2032.g.akamai.net/7/2032/5107/20021114070011/www.nortelnetworks%.com/solutions/ip_vpn/collateral/nn102260-110802.pdf)

Olavsrud, P. (2003). *Web services authentication takes leap forward.* (Available at http://www.internetnews.com/dev-news/article.php/ 1585771)

RSA. (2003). *Web Services Security*. (Available at http://www.rsasecurity. com/solutions/web-services/whitepapers/WSS_WP_0802.pdf)

Salz, R. (2003). *XKMS does the heavy work of PKI.* (Available at http://www.nwfusion.com/news/tech/2003/0908techupdate.html)

Siddiqui, B. (2002). *Exploring XML Encryption.* (Available at http://www-106.ibm.com/developerworks/xml/library/x-encrypt/)

Simon, E., Madsen, P., & Adams, C. (2001). *An Introduction to XML Digital Signatures.* (Available at http://www.xml.com/pub/a/2001/08/08/xmldsig.html)

SOAP. (2000, May). *Simple Object Access Protocol (SOAP) 1.1 Specification.* (Available at http://www.w3.org/TR/2000/NOTE-SOAP-20000508)

Stamps, P. (2003). *Presentation of an architecture for a message service handler.* (Available at http://www.softwareag.com/corporat/products/webservices/stamps.pdf)

UDDI. (2002, July). *Universal Description Discovery and Integration Technical Committee Specification Version 3.0.* (Available at http://uddi.org/pubs/uddi-v3.00-published-20020719.htm)

WS-Security Specification. (2004). *Web Services Security: SOAP Message Security 1.0.* (Available at http://www.verisign.com/wss/wss.pdf)

WSDL. (2001, Mrt). *Web Service Description Language WSDL 1.1 Specification.* (Available at http://www.w3.org/TR/2001/NOTE-wsdl-20010315)

XML Trust Centre. (2004). *Introduction to XKMS.* (Available at http://www.xmltrustcenter.org/xkms/index.htm)

The financial assistance of National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the National Research Foundation.